Machine Learning

Liyao Tang

March 4, 2019

Contents

List of Figures

List of Tables

Chapter 1

Math

1.1 Convolution

- Definition
  - $f * g(z) = \int_{\mathbb{R}} f(x)g(z-x)dx$, where $f(x), g(x)$ are functions in $\mathbb{R}$
- Statistical Meaning
  - Notation
    - X,Y: independent random variables, with pdf's given by f and g
    - $Z = X + Y$, with pdf given by $h(z)$:
  - $\Rightarrow h(z) = f * g(z)$
    - derivation

$$\begin{aligned} H(z) = P(Z < z) &= P(X + Y < z) \\ &= \int_x P(X = x)P(X + Y < z | X = x)dx \\ &= \int_x f(x)P(Y < z - x)dx \\ &= \int_x f(x)G_Y(z-x)dx \end{aligned}$$

$$\begin{aligned} \Rightarrow h(x) = \frac{d}{dz}H(z) &= \frac{d}{dz}\int_x f(x)G_Y(z-x)dx \\ &= \int_x f(x)\frac{dG_Y(z-x)}{dz}dx \\ &= \int_x f(x)g(z-x)dx \\ &= f * g(z) \end{aligned}$$

1.2 Linear Algebra

1.2.1 Essence

Vector

- Interpretation
  - Movement

        - direction
        - distance
    - Numeric in High Dimensions
        - in 1-D: +/- represents direction
        - in n-D: +/- alone each dimension combined to represent an overall direction (direction of the n-D numeric - vector)

- Numerics Multiplication
    - Scaling
        - the number scales the distance of vector (direction remains)
          $\Rightarrow$ such number thus also called scalar
        - $\Rightarrow$ scale alone each axis by that scalar
          $2\mathbf{x} = 2x_1e_2 + ... + 2x_ne_n$,
          where $e_1, ..., e_N$ are vector defining coordinates

- Linear Combination
    - Vector Adding: Generalization of Numerical Adding
        - in 1-D: joint movement along single axis
        - in n-D: joint movement along each axis $\Rightarrow$ a joint movement in n-D space
    - Definition: $\mathbf{x} = a_1\mathbf{x}_1 + ... + a_n\mathbf{x}_n$
        - the vectors $\mathbf{x}_1, ..., \mathbf{x}_n$ only altered linearly (as only being scaled)
          $\Rightarrow$ $\mathbf{x}$ direction & size are linear combination of that in $\mathbf{x}_1, ..., \mathbf{x}_n$
    - Span of $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$
        - the n-D space S constructed by linear combination of $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$
    - x_0 linearly dependent on $\{x_1, ., , , x_n\}$
        - x_0 can be constructed by linear combination of $\{x_1, ., , , x_n\}$
          (already in the span space S)
        - $\Leftrightarrow$ function $a_0\mathbf{x}_0 + ... + a_n\mathbf{x}_n = 0$ has other solution than $a_0 = ... = a_n = 0$
    - x_0 linearly INdependent with $\{x_1, ., , , x_n\}$
        - x_0 can NOT be constructed by linear combination of $\{x_1, ., , , x_n\}$
          (not in the span space S, will increase the dimension of S if adopted)
        - $\Leftrightarrow$ function $a_0\mathbf{x}_0 + ... + a_n\mathbf{x}_n = 0$ has and only has solution $a_0 = ... = a_n = 0$

- Special Vectors
    - n-D Zero Vector $\mathbf{x}$
        -
    - Unit Vector
    - Basis of Vector Space S^n
        - general basis: a set of linearly independent vectors that span the space
          (i.e. a set of linearly independent n-D vectors)
        - unit basis: a general basis where every vector is unit vector
        - orthogonal basis: a general basis where all vectors are orthogonal with each other
        - unit orthogonal basis: a basis that is also a unit basis and an orthogonal basis

    $\Rightarrow$ coordinate: the scaler to composite a vector given a specific basis

Linear Transformation and Maps

- Linear Transformation
  - Transformation
    - a function mapping: vector $\rightarrow$ vector
    - a vector movement: scale & rotate all possible input vectors (i.e. a vector space)
  - Transformation with Linearity $L(\cdot)$
    - intuition: lines remain lines & origin remains origin
    - definition: a transformation $L(\cdot)$ is linear if
      1. additivity: $L(\mathbf{x}_1 + \mathbf{x}_2) = L(\mathbf{x}_1) + L(\mathbf{x}_2)$
      2. scaling: $L(a\mathbf{x}) = aL(\mathbf{x})$, where a is scaler
  - Features of $L(\cdot)$ given $\mathbf{x} = x_1e_1 + ... + x_ne_n$
    - same scaler for coordinates
      $\Rightarrow L(\mathbf{x}) = L(x_1e_1 + ... + x_ne_n)$
      $= L(x_1e_1) + ... + L(x_ne_n)$
      $= x_1L(e_1) + ... + x_nL(e_n)$
      $\Rightarrow$ transformed vector $\mathbf{x}' = L(\mathbf{x})$ has the same coord under the transformed basis

- Linear Map
  - Definition
    - map $F : V \rightarrow X$ is a linear map if it is a linear transformation,
      where V, X are vector spaces

- Multilinear Maps
  - Definition
    - map $F : \underbrace{V \times ... \times V}_{k \text{ copies}} \rightarrow X$ is multilinear/k-linear if it is linear in each slot
      i.e. $F(\mathbf{v}_1, ..., a\mathbf{v}_i + b\mathbf{v}'_i, ..., \mathbf{v}_k) = aF(\mathbf{v}_1, ..., \mathbf{v}_i, ..., \mathbf{v}_k) + bF(\mathbf{v}_1, ..., b\mathbf{v}'_i, ..., \mathbf{v}_k)$
      i.e. for fixed $\mathbf{v}_1, ..., \mathbf{v}_{i-1}, \mathbf{v}_{i+1}, ..., \mathbf{v}_k, F$ reduced to linear map with $\mathbf{v}_i$ as variable
      (where V, X are vector spaces)
  - Alternating Maps
    - map F is alternating if, its output is $\mathbf{0}$ whenever two vectors in inputs are identical
  - for Multilinear Map F: F Alternating $\Leftrightarrow F(..., \mathbf{v}, ..., \mathbf{w}, ...) = -F(..., \mathbf{w}, ..., \mathbf{v}, ...)$
    i.e. for multilinear map F, F alternating $\Leftrightarrow$ swapping two inputs flips sign of output
    - proof: given multilinear and alternating map F, for any $\mathbf{v}, \mathbf{w}$
      $0 = F(..., (\mathbf{v} + \mathbf{w}), ..., (\mathbf{v} + \mathbf{w}), ...)$
      $= F(..., \mathbf{v}, ..., \mathbf{v}, ...) + F(..., \mathbf{w}, ..., \mathbf{w}, ...) + F(..., \mathbf{v}, ..., \mathbf{w}, ...) + F(..., \mathbf{w}, ..., \mathbf{v}, ...)$
      $= F(..., \mathbf{v}, ..., \mathbf{w}, ...) + F(..., \mathbf{w}, ..., \mathbf{v}, ...)$
      $\Rightarrow F(..., \mathbf{v}, ..., \mathbf{w}, ...) = -F(..., \mathbf{w}, ..., \mathbf{v}, ...)$
    - proof: given multilinear map $F : F(..., \mathbf{v}, ..., \mathbf{w}, ...) = -F(..., \mathbf{w}, ..., \mathbf{v}, ...)$
      $\Rightarrow F(..., \mathbf{v}, ..., \mathbf{v}, ...) = -F(..., \mathbf{v}, ..., \mathbf{v}, ...)$
      $\Rightarrow F(..., \mathbf{v}, ..., \mathbf{v}, ...) = 0$, hence alternating

Matrix

- Matrix for Linear Transformation $L : S \to S'$
  - Representing Space Transformation
    - package the transformed basis under the original basis using matrix M
      i.e. represent the $e'_1, ..., e'_n = L(e_1), ..., L(e_n)$ under the original basis $e_1, ..., e_n$
      $\Rightarrow M = [e'_1, ..., e'_n]$, with all transformed basis as column vectors
      $\Rightarrow M$ represent the result of linear transformation for the basis of S
    - hence, determine a linear space transformation $L : S \to S'$ using $e_1, ..., e_n$
      for $M_{m\times n}$ matrix: a linear transformation from n-D space to m-D space
      1. $m < n$: projecting to subspace
      2. $m > n$: expanding into a hyper-plane/-line/etc (constrained in hyper-space)
  - Performing Space Transformation
    - $\forall i \in \{1, ..., n\}, x'_i = i^{\text{th}}$ component of $\mathbf{x}' = L(\mathbf{x})$, then $x'_i = (e'_{1i} + ... + e'_{n_i})x_i$
      (as proved above)
    - $\Rightarrow$ output vector $\mathbf{x}' = L(\mathbf{x}) = M\mathbf{x}$ under the original basis $e_1, ..., e_n$
      hence the rule for matrix multiplication

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = x\begin{bmatrix} a \\ c \end{bmatrix} + y\begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} ax+by \\ cx+dy \end{bmatrix}$$

  where green and red columns the $\{e'_1, e'_2\}$ under $\{e_1, e_2\}$, $[x, y]$ the input vector $\mathbf{x}$

- Matrices for Composition of Linear Transformation
  - Transformation L_1, L_2 as Matrix M_1, M_2
    - as easy to prove $M_2 \cdot (M_1 \cdot \mathbf{x}) = (M_1 \cdot M_2) \cdot \mathbf{x}$
      $\Rightarrow L_2(L_1(\cdot)) \Leftrightarrow$ the composition of transformation defined by $M_2 \cdot M_1$
    - $\Rightarrow \mathbf{x}$ first transformed by L_1 then $L_2 \Leftrightarrow$ transformed by $M_2 \cdot M_1$
  - Linear Transformation on Space
    - given a basis matrix $E = [e_1, ..., e_n]$, a transformed basis $L_1(E) = L_1(e_1), ..., L_1(e_n)$
      ($e_1, ..., e_n$ as column vectors)
      $\Rightarrow L_2(L_1(e_k))$ performs L_2 transformation on the k^{th} vector of $L_1(E)$
    - hence to perform L_2 on the transformed space $L_1(E) \Rightarrow L_2(L_1(E))$
    - given that $L_1(E) = M_1 \Rightarrow L_2(L_1(E)) = M_2 \cdot M_1$
      (due to the derivation of linear transformation as matrix)
    - hence, $M_2 \cdot M_1$ denotes
      1. a further L_2 transformation on a transformed space $L_1(E)$
         (all result represented under the original basis E)
      2. the final transformed basis (first L_1 then L_2) under original basis E
         $\Rightarrow$ denotes a composite transformation of $L_2(L_1(\cdot))$
  - Multiplication between Matrices
    - $\Rightarrow$ composite linear transformation**s** into single linear transformation
    - $\Rightarrow$ linear transformation on vector**s**/space (generalized from single vector)
- Understanding Properties
  - $(AB)C = A(BC)$
    - both meaning apply transformation C then B then A...

$$\underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix}}^{M_2}\underbrace{\begin{bmatrix} e & f \\ g & h \end{bmatrix}}^{M_1}=\begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

where $M_1 = [L_1(e_1), L_1(e_2)], M_2 = [L_2(e_1), L_2(e_2)]$, the result $= [L_2(L_1(e_1)), L_2(L_1(e_2))]$ (all under basis $E = [e_1, e_2]$)

Dot Product

- Projecting to 1-D
  - Unit Orthogonal Basis
    - $\mathbf{x} \cdot \mathbf{y} = x_1y_1 + ... + x_ny_n = \mathbf{x}^T \cdot \mathbf{y} = \mathbf{y}^T \cdot \mathbf{x}$
      ($\mathbf{x}, \mathbf{y}$ assumed to be column vector / matrix with single column)
  - General Basis $E = [e_1, ..., e_n]$
    - $\Rightarrow \mathbf{x} = x_1e_1 + ... + x_ne_n = E\mathbf{x}, \mathbf{y} = y_1e_1 + ... + y_ne_n = E\mathbf{y}$
      $\Rightarrow \mathbf{x} \cdot \mathbf{y} = (E\mathbf{x}) \cdot (E\mathbf{y}) = \mathbf{x}^T E^T E\mathbf{y}$
    - understanding:
      1. $\mathbf{x} = E\mathbf{x}$: transfer back to a representation under unit orthogonal basis
      2. for $E = I$, back to the unit orthogonal case

  $\Rightarrow \mathbf{x} \cdot \mathbf{y}$: projecting $\mathbf{x}/\mathbf{y}$ to 1-D line using transformation $\mathbf{y}/\mathbf{x}$
  (direction/scaling effect of projection can be alter by choice of basis E though)

- Duality
  - Dual Vector
    - the vector represent a projection (linear transformation) to 1-D line
      (hence the vector equivalent to the matrix with 1 row defining the projection)
    - $\Rightarrow$ performing transformation on a vector $\Leftrightarrow$ taking product with the dual vector

Exterior (Wedge) Product

- Introduction
  - Definition
    - kth exterior product $\wedge^k V$ is a vector space, with a map $\psi : \underbrace{V \times ... \times V}_{k \text{ times}} \to \wedge^k V$
      note: map ψ called exterior multiplication, element $\psi(\mathbf{v}_1 \wedge ... \wedge \mathbf{v}_k)$ called k-blade
  - Key Properties for Pair $(\wedge^k V, \psi)$
    - ψ is alternating multilinear map
    - for basis $\{e_1, ..., e_n\}$ of $V \in \mathbb{R}^n \Rightarrow \{e_{i_1} \wedge ... \wedge e_{i_k} | 1 \leq i_1 \leq i_k \leq n\}$ a basis for $\wedge^k V$
      (due to its alternating multilinearity)
      $\Rightarrow e_{i_1} \wedge ... \wedge e_{i_k}$ can form any permutation of the same input by swapping order
      e.g. for $\wedge^2 V : e_1 \wedge e_1 = e_2 \wedge e_2 = \mathbf{0}, e_1 \wedge e_2 = -e_2 \wedge e_1 \Rightarrow$ not linearly independent
    - dim $\wedge^k V = \binom{n}{k} = C_n^k$ (due to the form of its basis),
      where $n =$ dim V
    - sum of k-wedge is still in the vector space $\wedge^k V$
    - any alternating multilinear map $F : \underbrace{V \times ... \times V}_{k} \to X$ factors uniquely into:

      $\underbrace{V \times ... \times V}_{k} \xrightarrow{\psi} \wedge^k V \xrightarrow{\underline{F}} X$, where ψ exterior multiplication, $\underline{F}$ a linear map

Determinant

- Signed Volume
-
- Measuring Linear Transformation on Volume
  - Unit Volume
    - unit volume $v = \|e_1 \bigwedge ... \bigwedge e_n\|$ with basis $e_1, ..., e_n$
      (for orthogonal basis, $v = \|e_1\| \times ... \times \|e_n\|$)
    - after transformation: $L(v) = \|L(e_1) \bigwedge ... \bigwedge L(e_n)\| = \|Me_1 \bigwedge ... \bigwedge Me_n\|$

    where $\bigwedge$ is the exterior product
  - Measuring Change of Unit Volume
    - $\Rightarrow \det(M) = \frac{L(v)}{v} = \|M_0 \times ... \times M_n\|$,
      assuming unit orthogonal basis $E = I$, where I is identity matrix
      (note: interpretation of $\det(\cdot) \leftrightarrow$ spacial interpretation of cross product $\times$)
    - hence, the rule of calculating $\det(\cdot)$
- Measuring Change of Orientation
  - Orientation of Space
    - jointly defined by the direction & order of the sequence of basis vector
      i.e. the positive/negative part of each axis in sequence
  - Measuring the Change
    - $\det(M) < 0$ if axises flipped over once (for an odd times)
      $\det(M) > 0$ if flipped for even times
    - interpretation: the flipped axis approaches 0 then expanded into the negative
      (measured by original basis E)
- Linear Dependency
  - $\det(M) = 0$
    - volume in current space becomes 0
      $\Rightarrow$ dimensions decreases after transformation applied
      $\Rightarrow$ i.e. transformed basis not able to span the current space
    - $\Rightarrow$ basis in M NOT linearly INdependent! $\Leftrightarrow \det(M) = 0$
  - $\det(M) \neq 0$
    - volume still exist
      $\Rightarrow$ dimensions remain & transformed basis still span the space
    - $\Rightarrow$ basis in M is linearly INdependent $\Leftrightarrow \det(M) \neq 0$
- Understanding Properties
  - $\det(M_1M_2) = \det(M_1)\det(M_2)$:
    - left: final volume & orientation changed after transformation M_2 then M_1
    - right: the volume scaled by one transformation, then further by the other;
      (similar for orientation, as measured by sign)

Cross Product

- Determinant
  - $\|\mathbf{v} \times \mathbf{w}\| = |\det([\mathbf{v}, \mathbf{w}])|$
    - as determinant measuring the change of unit basis
      $\Rightarrow$ constructing matrix with target vector $\mathbf{v}, \mathbf{w}$ as column
    - $\Rightarrow$ matrix $[\mathbf{v}, \mathbf{w}]$ as the transformation altering the space
  - Direction of $\mathbf{v} \times \mathbf{w}$
    - expanding in the perpendicular direction w.r.t the hyper-plane defined by $\mathbf{v}, \mathbf{w}$ (obeying the "right hand rule")
- Linear Transformation
  -

Rank

- Measuring Linear Transformation on Dimension
  - Measuring the Transformed Space
    - rank of $M = r$: basis in M span a r-dimension (column) space
      note: the column space of M: transformed space defined by column basis
    - full rank: the dimension of column space is as high as possible
- Null Space (Kernel) of M
  - Vectors in Null Space
    - $\forall \mathbf{x}, M\mathbf{x} = \mathbf{0}$
      (i.e. all the vectors in null space transformed into $\mathbf{0}$ by M, hence the name)
    - $\mathbf{0}$ always in null space
  - Dimension of Null Space
    - $D(\text{null space}) = \text{num of column in } M - \text{rank of } M$
      note: as focusing on column space $\Rightarrow$ num of cols = dimension of input vectors
- Understanding Properties
  - $R(M_{m\times n}) \le \min\{m, n\}$
    - $m < n$: projecting a n-D vector to a m-D subspace, hence at most of rank m
    - $m > n$: M consists of n vectors of m dimensions, define at most n-D space
  - $R(M_{m\times n}) = \min\{m, n\} \Leftrightarrow M$ Full Rank
    - from definition, it is as high as possible
    - m vectors with n dimensions, span at most a $\min\{m, n\}$ space
      either linearly dependent ($m < n$), or not enough independent vectors ($m > n$)

Inverse of Matrix

- Inverse of Linear Transformation
  - Interpretation
    - a transformation L^{-1} to inverse the effect of another transformation L
      $\Rightarrow \forall \mathbf{x}, L^{-1}L(\mathbf{x}) = \mathbf{x}$
    - $\Rightarrow M^{-1}M = I$, where M for L, M^{-1} for L^{-1}, I the identity matrix

- Requirement
  - transformed basis in M still span the space!
    otherwise, transformed vectors projected onto subspace $\Rightarrow$ information lost!
  - hence, following equivalent requirements:
    1. transformed basis in M linearly INdependent
    2. M is square and full rank
    3. $\det(M) \neq 0$

Linear System of Equations

- Linearity
  - Linear Combination of Variables
    - coefficients matrix A: holding the coefficient for each equation (in row)
    - variables vector $\mathbf{x}$: holding variables as column vector
    - constants vector $\mathbf{v}$: holding target constant for each equations as column vector

    $\Rightarrow A\mathbf{x} = \mathbf{v}$ for a set of linear equations
  - Linear Transformation Perspective
    - $\mathbf{x}/\mathbf{v}$ as original/transformed vectors
    - columns of A (coefficients for the same variable) as transformed basis

    $\Rightarrow$ finding a start position $\mathbf{x}$ which, after transformation A, lands on $\mathbf{v}$
- Solutions
  - Transformed Basis Linearly INdependent ($\det(A) \neq 0$)
    - hence, $A\mathbf{x} = \mathbf{v} \Leftrightarrow \mathbf{x} = A^{-1}\mathbf{v}$
      (use the inverse transformation A^{-1} to find the input $\mathbf{x}$ using output $\mathbf{v}$)

    $\Rightarrow$ single unique $\mathbf{x}$ found
  - Transformed Basis Linearly Dependent ($\det(A) = 0$)
    - information lost ($\mathbf{x}$ projected to subspace) $\Rightarrow$
      1. multiple solutions: basis in A linearly dependent with $\mathbf{v}$
         $\Rightarrow$ i.e. $\mathbf{v}$ in the column space of A
         (special case where $\mathbf{v} = \mathbf{0}$: solution space = null space of A)
      2. no solution: basis in A linearly INdependent with $\mathbf{v}$
         $\Rightarrow$ i.e. can NOT possibly be described by basis in A

1.2.2 Interchanging Coordinates

N-dimensional Spherical Coordinates

- Notation
  - N-dim Euclidean Space E_N
    - $e_1, e_2, ..., e_N$: a group of orthonormal basis of E_N
    - $\mathbf{x} = (x_1, x_2, ..., x_N)$: vector in E_n
    - $\mathbf{x}_{i-N}$: projection of $\mathbf{x}$ onto subspace spanned by $e_i, ..., e_N$
      $\Rightarrow \mathbf{x}_{i-N} = \sum_{n=i}^{N} x_n e_n$
  - Spherical Coordinates

    - $r = \|\mathbf{x}\|$: the norm of $\mathbf{x}$
    - $\phi_i \in [0, \pi]$: angle between $\mathbf{x}_{i-N}$ and e_i
    - $r_i = \|\mathbf{x}_{i-N}\|$: norm of projection $\mathbf{x}_{i-N}$, with $r_1 = r$

- Observation
  - Space $e_1, ..., e_N$:
    - $\cos\phi_1 = \frac{\mathbf{x}e_1}{\|\mathbf{x}\|\|e_1\|} = \frac{x_1}{r}$
      $\Rightarrow x_1 = r\cos\phi_1$
      $$\Rightarrow \mathbf{x} = r\cos\phi_1 e_1 + \sum_{n=2}^{N} x_n e_n$$
  - Space $e_2, ..., e_N$:
    - from above: $\mathbf{x}^2 = r^2\cos^2\phi_1 + \sum_{n=2}^{N} x_n^2 = r^2$
      $$\Rightarrow \sum_{n=2}^{N} x_n^2 = r^2\sin^2\phi_1$$
    - $\mathbf{x}_{2-N} = \sum_{n=2}^{N} x_n e_n$
      $$\Rightarrow \begin{cases} \|\mathbf{x}_{2-N}\|^2 = \sum_{n=2}^{N} x_n^2 = r^2\sin^2\phi_1 = r_2^2 \\ \cos\phi_2 = \dfrac{\mathbf{x}_{2-N}\cdot e_2}{\|\mathbf{x}_{2-N}\|\|e_2\|} = \dfrac{x_2}{r_2} \end{cases}$$
      $$\Rightarrow \begin{cases} r_2 = r\sin\phi_1 & (\text{ as } \phi_1 \in [0, \pi]) \\ x_2 = r_2\cos\phi_2 = r\sin\phi_1\cos\phi_2 \end{cases}$$
      $$\Rightarrow \mathbf{x}_{2-N} = r\sin\phi_1\cos\phi_2 e_2 + \sum_{n=3}^{N} x_n e_n$$
  - Space $e_3, ..., e_N$:
    - from above: $\mathbf{x}_{2-N}^2 = r^2\sin^2\phi_1\cos^2\phi_2 + \sum_{n=3}^{N} x_n^2 = r^2\sin^2\phi_1$
      $$\Rightarrow \sum_{n=3}^{N} x_n^2 = r^2\sin^2\phi_1\sin^2\phi_2$$
    - $\mathbf{x}_{3-N} = \sum_{n=3}^{N}$
      $$\Rightarrow \begin{cases} \|\mathbf{x}_{3-N}\|^2 = \sum_{n=3}^{N} x_n^2 = r^2\sin^2\phi_1\sin^2\phi_2 = r_3^2 \\ \cos\phi_3 = \dfrac{\mathbf{x}_{3-N}\cdot e_3}{\|\mathbf{x}_{3-N}\|\|e_3\|} = \dfrac{x_3}{r_3} \end{cases}$$
      $$\Rightarrow \begin{cases} r_3 = r\sin\phi_1\sin\phi_2 \quad (\text{ as } \phi_1, \phi_2 \in [0, \pi]) \\ x_3 = r_3\cos\phi_3 \end{cases}$$
      $$\Rightarrow \mathbf{x}_{3-N} = r\sin\phi_1\sin\phi_2 e_3 + \sum_{n=4}^{N} x_n e_n$$
- Proof for x_i
  - Procedure

- $\mathbf{x}_{i-N} = \sum_{n=i}^{N} x_n e_n$
  $\Rightarrow \cos\phi_i = \frac{\mathbf{x}_{i-N} \cdot e_i}{\|\mathbf{x}_{i-N}\| \|e_i\|} = \frac{x_i}{r_i}$
  $\Rightarrow x_i = r_i \cos\phi_i$

- Induction
  - Goal
    - $\forall i \geq 2, r_i = r \prod_{j=1}^{i-1} \sin\phi_j$
  - Base Case ($i = 2$)
    - as in observation, $r_2 = r \sin\phi_1 = r \prod_{j=1}^{2-1} \sin\phi_j$
  - Step Case
    - assumption $r_i = r \prod_{j=1}^{i-1} \sin\phi_j$
    - procedure: $\mathbf{x}_{i-N} = \sum_{n=i}^{N} x_n e_n = r_i \cos\phi_i + \sum_{n=i+1}^{N} x_n e_n$
      $\Rightarrow \|x_{i-N}\|^2 = r_i^2 \cos^2\phi_i + \sum_{n=i+1}^{N} x_n^2 = r_i^2$
      $\Rightarrow \|x_{i+1-N}\|^2 = \sum_{n=i+1}^{N} x_n^2 = r_i^2 \sin^2\phi = r_{i+1}^2$
      $\Rightarrow r_{i+1} = r_i \sin\phi_i = r \prod_{j=1}^{i} \sin\phi_j$

- Derivation
  - x_i from Combined Proofs
    - $x_i = \begin{cases} r\cos\phi_1 & i = 1 \\ r\cos\phi_i \prod_{j=1}^{i-1} \sin\phi_j & 2 \leq i \leq N \end{cases}$
  - Last 2 Dimensions
    - $\mathbf{x}_{(N-1)-N} = x_{N-1} \cdot e_{N-1} + x_N \cdot e_N$, with $r_{N-1} = r \prod_{j=1}^{N-2} \sin\phi_j$
      $\Rightarrow \|\mathbf{x}_{(N-1)-N}\| = f(\phi_{N-1}, \phi_N) = r_{N-1}$
      $\Rightarrow \phi_{N-1}, \phi_N$ not independent**!**
      (actually, if $e_N = e_{N-1} + \frac{\pi}{2}$, then $\phi_N = \phi_{N-1} - \frac{\pi}{2}$)

      $\Rightarrow$ define $\theta \in [0, 2\pi)$ instead of $\phi_{N-1}, \phi_N \in [0, \pi]$
      $\Rightarrow x_{N-1} = r_{N-1} \sin\theta, x_N = r_{N-1} \cos\theta$ (interchangeable)
  - Final Spherical Coordinates

- $x_i = \begin{cases} r\cos\phi_1 & i=1 \\ r\cos\phi_i \prod_{j=1}^{i-1}\sin\phi_j & 2\le i\le N-1 \\ r\sin\theta \prod_{j=1}^{N-2}\sin\phi_j & i=N-1 \\ r\cos\theta \prod_{j=1}^{N-2}\sin\phi_j & i=N \end{cases}$

1.3 Calculus

1.3.1 Integral

Interchanging Coordinates in Integral

- General Theory
  - Notation
    - (x, y): coordinate under Field D
    - (u, v): coordinate under Field D'
    - $T: \begin{cases} x = x(u,v), \\ y = y(u,v) \end{cases}$: transformation from D to D'
  - Assumption
    - $f(x, y)$ continuous in D
    - transformation T's partial 1^{st} order derivatives continuous on D'
    - transformation T's Jacobian $J(u,v) = \frac{\partial(x,y)}{\partial(u,v)} \neq 0$
    - transformation $T : D \to D'$ is 1-1 mapping
  - Derivation
    - take infinitely small square in D' :
      $M_4'(u, v+\delta v)$, $M_3'(u+\delta u, v+\delta v)$,
      $M_1'(u, v)$, $M_2'(u+\delta u, v)$

      $\Rightarrow$ after transformation to D :
      $M_4(x(u, v+\delta v), y(u, v+\delta v))$, $M_3(x(u+\delta u, v+\delta v), y(u+\delta u, v+\delta v))$,
      $M_1(x(u,v), y(u,v))$, $M_2(x(u+\delta u, v), y(u+\delta u, v))$

      $$\Rightarrow x_2 - x_1 = x(u+\delta u, v) - x(u,v) = \frac{\partial x}{\partial u}|_{(u,v)}\delta u$$
      $$x_4 - x_1 = x(u, v+\delta v) - x(u,v) = \frac{\partial x}{\partial v}|_{(u,v)}\delta v$$
      $$y_2 - y_1 = y(u+\delta u, v) - y(u,v) = \frac{\partial y}{\partial u}|_{(u,v)}\delta u$$
      $$y_4 - y_1 = y(u, v+\delta v) - y(u,v) = \frac{\partial y}{\partial v}|_{(u,v)}\delta v$$

      as $\delta u, \delta v \to 0$, curvilinear boundary quadrilateral $M_1M_2M_3M_4 \to$ parallelogram
      $$\Rightarrow S_{M_1M_2M_3M_4} = |\overrightarrow{M_1M_2} \times \overrightarrow{M_1M_4}| = |\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_4 - x_1 & y_4 - y_1 \end{vmatrix}|$$
      $$= |\begin{vmatrix} \frac{\partial x}{\partial u}\delta u & \frac{\partial y}{\partial u}\delta u \\ \frac{\partial x}{\partial v}\delta v & \frac{\partial y}{\partial v}\delta v \end{vmatrix}| = |\begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} \end{vmatrix}|\delta u\delta v$$
      $$= |J(u,v)|\delta u\delta v$$

- $\Rightarrow$ infinitely small area $\delta\sigma = dxdy = |J(u,v)|\delta u\delta v$
  $\Rightarrow \int\int_D f(x,y)dxdy = \int\int_{D'} f(x(u,v),y(u,v))|J(u,v)|dudv$

- Integral in Cartesian $\rightarrow$ Polar
  - Result
    - $dxdy = rdrd\theta$
  - Derivation
    - from general transformation: $x = r\cos(\theta), y = r\sin(\theta)$
      $\Rightarrow dxdy = |J(r,\theta)|drd\theta = rdrd\theta$
    - from direct calculation of infinite small area in polar coordinate
      $\Rightarrow d\sigma = \frac{1}{2}(r+dr)^2d\theta - \frac{1}{2}r^2d\theta = rdrd\theta + \frac{1}{2}(dr)^2d\theta$
      $\Rightarrow d\sigma = rdrd\theta, \text{when } dr, d\theta \rightarrow 0$

Gaussian Integral

- Gaussian Function
  - $f(x) = \mathrm{e}^{-\mathrm{a(x+b)}^2}$
    - special form: $f(x) = \mathrm{e}^{-\mathrm{(x)}^2}$
    - alternative form: $f(x) = \mathrm{e}^{\mathrm{ax^2+bx+c}}$
    - no indefinite integral $\int_a^b \mathrm{e}^{-\mathrm{x}^2}$
    - only definite integral $\int_{-\infty}^{+\infty} \mathrm{e}^{-\mathrm{x}^2}$
- Direct Integral
  - $$\begin{aligned}(\int_{-\infty}^{+\infty} \mathrm{e}^{-\mathrm{a(x+b)}^2}\mathrm{dx})^2 &= \int_{-\infty}^{+\infty} \mathrm{e}^{-\mathrm{a(x+b)}^2}\mathrm{dx}\int_{-\infty}^{+\infty} \mathrm{e}^{-\mathrm{a(y+b)}^2}\mathrm{dy} \\ &= \int\int_{-\infty}^{+\infty} \mathrm{e}^{-\mathrm{a[(x+b)^2+(y+b)^2]}}\mathrm{d(x+b)d(y+b)} = \int\int_{-\infty}^{+\infty} \mathrm{e}^{-\mathrm{a(x^2+y^2)}}\mathrm{dxdy} \\ &= \int_0^{2\pi}\int_0^{+\infty} \mathrm{e}^{-\mathrm{ar}^2}\mathrm{rdrd}\theta \\ &= \frac{\pi}{a}\end{aligned}$$
  $$\Rightarrow \int_{-\infty}^{+\infty} \mathrm{e}^{-\mathrm{a(x+b)}^2}\mathrm{dx} = \sqrt{\frac{\pi}{\mathrm{a}}}, \text{ alternatively } \int_{-\infty}^{+\infty} \mathrm{e}^{\mathrm{ax^2+bx+c}}\mathrm{dx} = \sqrt{\frac{\pi}{-\mathrm{a}}}\cdot \mathrm{e}^{\frac{\mathrm{b}^2}{4\mathrm{a}}+\mathrm{c}}$$
- Even Moment of Gaussian Function
  - $$\begin{aligned}\int_{-\infty}^{+\infty} x^{2n}\mathrm{e}^{-\mathrm{ax}^2}\mathrm{dx} &= (-1)^n\int_{-\infty}^{+\infty}\frac{\partial^n}{\partial a^n}\mathrm{e}^{-\mathrm{ax}^2}\mathrm{dx} \\ &= (-1)^n\frac{\partial^n}{\partial a^n}\int_{-\infty}^{+\infty}\mathrm{e}^{-\mathrm{ax}^2}\mathrm{dx} && \text{by parameter differeation} \\ &= (-1)^n\sqrt{\pi}\frac{\partial^n}{\partial a^n}a^{-\frac{1}{2}} \\ &= \sqrt{\frac{\pi}{a}}\frac{(2n-1)!!}{(2a)^n}, && \text{where !! is double factorial}\end{aligned}$$

1.4 Probability Theory

1.4.1 Introduction

Background

- Measuring Uncertainty
  - Source of Uncertainty
    - noise in reality & observation
    - finite size of data (limited information)
- Derivation
  - Quantifying Belief
    - by Cox (1946): if numerical values used to represent degrees of belief, a simple set of axioms encoding common sense properties of such beliefs will lead uniquely to a set of rules for manipulating degrees of belief that are equivalent to the sum and product rules of probability
  - Measuring Uncertainty
    - by Jaynes (2003): probability theory can be regarded as an extension of Boolean logic to situations involving uncertainty
  - Common Destination
    - numerical quantities to measure uncertainty, derived from different sets of properties/axioms, behave precisely according to the rules of probability

The Basic

- Notation
  - X, Y: random variable
- Discrete
  - $P(X, Y)$: joint probability of X, Y taking their values
  - $P(X)$: marginal probability of X taking its value
  - $P(X|Y)$: conditional probability of X taking its value given Y observed / determined
- Continuous
  - $P(x) = P_X(x)$: cumulative probability of value for variable $X < x$
  - $p(x)$: probability density,
    - where $\lim\limits_{\delta x \to 0} P(X \in (x, x+\delta x)) = \lim\limits_{\delta x \to 0} p(x)\delta x \Rightarrow P(X \in (a,b)) = \displaystyle\int_a^b p(x)dx$
    - $\Rightarrow p(x) \geq 0$ and $\displaystyle\int_{-\infty}^{+\infty} p(x) = 1$
    - $\Rightarrow P(z) = \int_{-\infty}^{z} p(x)dx$
- Basic Rules
  - Sum Rule
    - $P(X) = \displaystyle\sum_Y P(X, Y)$, where X, Y are discrete

- $P(X) = \int_Y P(X,Y)$, where X, Y are continuous
  (formal justification requires measure theory)
- Product Rule
  - $P(X,Y) = P(Y|X)P(X)$
  - $P(X,Y) = P(Y)P(X)$, where X, Y are independent
- $\Rightarrow$ Bayes' Rule
  - $P(Y|X) = \dfrac{P(X|Y)P(Y)}{P(X)} = \dfrac{P(X|Y)P(Y)}{\sum_Y P(X|Y)P(Y)}$, where Y are discrete
  - $P(Y|X) = \dfrac{P(X|Y)P(Y)}{P(X)} = \dfrac{P(X|Y)P(Y)}{\int_Y P(X|Y)P(Y)}$, where Y are continuous

- Interpretation of Bayes
  - Normalization
    - the $\sum$, $\int$ can be interpreted as a **normalization constant**
      $\Rightarrow$ **posterior** $\propto$ **likelihood** $\times$ **prior**
  - Prior
    - $P(Y)$: available probability of desired variable **before** anything observed
      $\Rightarrow$ Y usually model parameters
  - Posterior
    - $P(Y|X)$: obtained probability of desired variable **after** observation
      $\Rightarrow$ if X, Y independent, observation has no effect $\Rightarrow$ prior = posterior
  - Likelihood
    - $P(X|Y)$: how probable/likely of X being observed under different setting of Y
  - Prior $\rightarrow$ Posterior
    - a process of incorporating the evidence provided by observation

1.4.2 Expectations and Covariances

Expectation

- Definition
  - Expectation of $f(x)$ under $p(x)$
    - discrete x: $\mathbb{E}_p[f] = \sum_x p(x)f(x)$
    - continuous x: $\mathbb{E}_p[f] = \int p(x)f(x)dx$
    - approximation with N points drawn from $p(x)$: $\mathbb{E}_p[f] \simeq \dfrac{1}{N}\sum_{n=1}^{N} f(x_n)$
      (when $N \rightarrow \infty$, $\simeq$ becomes $=$)
  - Multivariate Expectation
    - Marginal Expectation of $f(x,y)$ on x: $\mathbb{E}_x[f(x,y)] = \sum_x p(x)f(x,y)$
      (hence a function of y)
    - Conditional Expectation $f(x)$ on $p(x|y)$: $\mathbb{E}[f|y] = \sum_x p(x|y)f(x)$

- Independence

- Independent x, y
  - $\mathbb{E}_{xy}[x, y] = \sum_{x,y} p(x, y)xy = \sum_{x,y} p(x)p(y)xy = \mathbb{E}[x]\mathbb{E}[y]$

Variance

- Definition
  - Variance of $f(x)$:
    - $\mathrm{var}[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2]$
      $= \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$
  - Covariance

Covariance

- Definition
  - between Variables x, y
    - $\mathrm{cov}[x, y] = \mathbb{E}_{x,y}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])]$
      $= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y]$
  - between Vectors $\mathbf{x}, \mathbf{y}$ (column vectors)
    - $\mathrm{cov}[\mathbf{x}, \mathbf{y}] = \mathbb{E}_{\mathbf{x},\mathbf{y}}[(\mathbf{x} - \mathbb{E}[\mathbf{x}]) \cdot (\mathbf{y}^T - \mathbb{E}[\mathbf{y}^T])]$
      $= \mathbb{E}_{\mathbf{x},\mathbf{y}}[\mathbf{x}\mathbf{y}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^T]$
      (pairwise covariance between components of $\mathbf{x}, \mathbf{y}$)
  - within Vector $\mathbf{x}$
    - $\mathrm{cov}[\mathbf{x}] \equiv \mathrm{conv}[\mathbf{x}, \mathbf{x}]$
      (pairwise covariance between its components)
- Independence Variable
  - Independent x, y
    - $\mathrm{cov}[x, y] = \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y] = 0$

1.4.3 Transformations of Random Variables

Inverse Image

- Definition
  - Notation
    - function $g : \mathbb{R} \to \mathbb{R}$
    - set A in $\mathbb{R}$
  - Inverse Image on Set A
    - $g^{-1}(A) = \{x \in \mathbb{R} | g(x) \in A\}$
      $\Leftrightarrow x \in g^{-1}(A)$ if and only if $g(x) \in A$
      interpretation: for each element in A, get its original value before g applied
- Properties
  - $g^{-1}(\mathbb{R}) = \mathbb{R}$, as g is defined on $\mathbb{R}$
  - $\forall$ set $A, g^{-1}(A^c) = g^{-1}(A)^c$, where A^c is the complement of set A

  - $\forall$ collection of sets $\{A_\lambda | \lambda \in \Lambda\}$, $g^{-1}\left(\bigcup_\lambda A_\lambda\right) = \bigcup_\lambda g^{-1}(A_\lambda)$
  - General Transformation $Y = g(X)$
    - $P(Y \in A) = P(g(X) \in A) = P(X \in g^{-1}(A))$

Discrete Variable

- Variable
  - X: random variable with probability mass function $P_X(x)$
  - $Y = g(X)$, with probability mass function $P_Y(y)$
- Probability Mass Function
  - $P_Y(y) = \sum_{x \in g^{-1}(y)} P_X(x)$, as $X = x$ is independent and mutually exclusive
    note: $g^{-1}(y)$ denotes $g^{-1}(\{y\})$
  - Example
    - uniform random variable X on $\{-n, ..., n\}$ with $Y = |X|$
      $\Rightarrow P_X(x) = \frac{1}{2n+1}$
      $\Rightarrow P_Y(y) = \begin{cases} \frac{1}{2n+1}, & x = 0, \\ \frac{2}{2n+1}, & x \neq 0. \end{cases}$

Continuous

- Variable
  - X: random variable with cumulative distribution $P_X(x)$, density $p_X(x)$
  - $Y = g(X)$, with cumulative distribution $P_Y(y)$, density $p_Y(y)$
- Cumulative Distribution
  - Strictly Monotone Increasing g
    - $P_Y(y) = P(Y \leq y) = P(g(X) \leq y) = P(X \leq g^{-1}(y)) = P_X(g^{-1}(y))$
  - Strictly Monotone Decreasing g
    - $P_Y(y) = P(Y \leq y) = P(g(X) \leq y) = P(X \geq g^{-1}(y)) = 1 - P_X(g^{-1}(y))$
- Probability Density
  - Strictly Monotone g (an one-to-one function)
    - $p_Y(y) = \frac{d}{dy} P_Y(y) = \frac{dP_Y(y)}{dg^{-1}(y)} \frac{dg^{-1}(y)}{dy} = p_X(g^{-1}(y)) \left| \frac{d}{dy} g^{-1}(y) \right|$,
      as g^{-1} has the same monotony as g, combined with the sign in P_Y to give the $|\cdot|$

1.4.4 Gaussian Distribution

Definition

- Univariate Gaussian
  - Variable
    - mean: μ
    - variance: $\sigma^2 \Rightarrow$ reciprocal of the variance $\beta = \frac{1}{\sigma^2}$ (also called precision)

- Probability Dense Function (PDF)
  - $\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$
    $\Rightarrow$ satisfying probability axioms: $\mathcal{N}(x|\mu,\sigma^2) > 0$ and $\int_{-\infty}^{+\infty} \mathcal{N}(x|\mu,\sigma^2)dx = 1$
- Expectation
  - $\mathbb{E}[x] = \int_{-\infty}^{+\infty} \mathcal{N}(x|\mu,\sigma^2)xdx = \mu$

$$
\begin{aligned}
\Rightarrow \mathbb{E}[x^2] &= \int_{-\infty}^{+\infty} \mathcal{N}(x|\mu,\sigma^2)x^2dx \\
&= \frac{1}{(2\pi\sigma^2)^{1/2}} \int_{-\infty}^{+\infty} x^2 \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} dx \\
&= \pi^{-\frac{1}{2}} \int_{-\infty}^{+\infty} (\sqrt{2\sigma^2}x+\mu)^2 \exp(-x^2)dx,\ \text{substituting } a = \frac{x-\mu}{\sqrt{2\sigma^2}} \\
&= \pi^{-\frac{1}{2}}(2\sigma^2 \int_R x^2\mathrm{e}^{-\mathrm{x}^2}\mathrm{dx} + 2\mu\sqrt{2\sigma^2}\int_{\mathrm{R}} \mathrm{xe}^{-\mathrm{x}^2}\mathrm{dx} + \mu^2 \int_{\mathrm{R}} \mathrm{e}^{-\mathrm{x}^2}\mathrm{dx}) \\
&= \pi^{-\frac{1}{2}}(2\sigma^2 \int_R x^2\mathrm{e}^{-\mathrm{x}^2}\mathrm{dx} + 2\mu\sqrt{2\sigma^2}\cdot 0 + \mu^2\sqrt{\pi}) \\
&= 2\sigma^2\pi^{-\frac{1}{2}} \int_R x^2\mathrm{e}^{-\mathrm{x}^2}\mathrm{dx} + \mu^2 \\
&= \sigma^2 + \mu^2,\ \text{by 2nd moment of Guassian or } (x\mathrm{e}^{-\mathrm{x}^2})' = \mathrm{e}^{-\mathrm{x}^2} - 2\mathrm{x}^2\mathrm{e}^{-\mathrm{x}^2}
\end{aligned}
$$

- Variance
  - $\mathrm{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2$

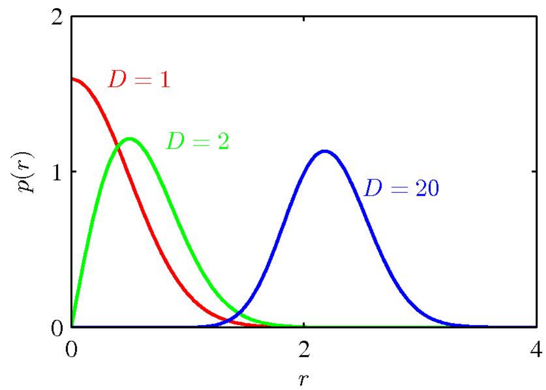
- Multivariate (d-dimensional) Gaussian
  - Variable
    - mean: $\mu \in \mathbb{R}^d$
    - covariance matrix: $\Sigma_{d\times d}$
  - Probability Dense Function (PDF)
    - $\mathcal{N}_d(\mathbf{x}|\mu,\mathbf{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$,
      noted as $X \sim \mathcal{N}_d(x|\mu,\Sigma)$

Multivariate Gaussian

- Dimensionality
  - Volume of High Dimensional Sphere
    - for $n = 2k, k \in N^+, V_{2k}(R) = \frac{\pi^k}{k!}R^{2k}$
    - for $n = 2k+1, k \in N, V_{2k}(R) = \frac{2^{k+1}\pi^k}{(2k+1)!!}R^{2k+1}$
    - $\Rightarrow \lim_{D\to+\infty} \frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = \lim_{D\to+\infty} 1 - 1(1-\epsilon)^D = 1 \Rightarrow$ the volume of a D-sphere concentrate in a thin shell near the surface!
      (actually, in the corner of a high dimensional cube as shown below)
  - Volume of High Dimensional Cube
    -

  - $\Rightarrow$ volume ratio of hyper sphere and hyper cube: $\Rightarrow$ the volume of a D-cube concentrates in its corner! $\Rightarrow$ distance function in high dimensional space CAN be useless

- High Dimensional Distribution
- High Dimensional Gaussian
  - probability density with respect to radius r for various dimension D
    $\Rightarrow$ most density are in a thin shell at a specific r



- Facing High Dimensionality

* Convolution of Gaussian
  - Integral of Gaussians $\int G_1 G_2 dx$
    - $G_1 \sim \mathcal{N}_d(x|a, A), G_2 \sim \mathcal{N}_d(x|b, B)$

$$\begin{aligned}
&\Rightarrow \int \mathcal{N}_d(x|a, A) N_d(x|b, B) dx \\
&= \int \frac{1}{(2\pi)^{d/2}|A|^{1/2}} e^{-\frac{1}{2}(x-a)^T A^{-1}(x-a)} \frac{1}{(2\pi)^{d/2}|B|^{1/2}} e^{-\frac{1}{2}(x-b)^T B^{-1}(x-b)} dx \\
&= \int \frac{1}{(2\pi)^{d/2}|A|^{1/2}} \frac{1}{(2\pi)^{d/2}|B|^{1/2}} e^{-\frac{1}{2}[(x-a)^T A^{-1}(x-a)+(x-b)^T B^{-1}(x-b)]} \\
&= \int \frac{1}{(2\pi)^{d/2}|A|^{1/2}} \frac{1}{(2\pi)^{d/2}|B|^{1/2}} e^{-\frac{1}{2}[(x-c)^T (A^{-1}+B^{-1})(x-c)+(a-b)^T C(a-b)]}, \\
&\quad \text{where } c = (A^{-1}+B^{-1})^{-1}(A^{-1}a + B^{-1}b), C = A^{-1}(A^{-1}+B^{-1})^{-1}B^{-1} = (A+B)^{-1} \\
&= \frac{\left|(A^{-1}+B^{-1})^{-1}\right|^{1/2}}{(2\pi)^{d/2}|A|^{1/2}|B|^{1/2}} e^{-\frac{1}{2}(a-b)^T C(a-b)} \int \frac{1}{(2\pi)^{d/2}\left|(A^{-1}+B^{-1})^{-1}\right|^{1/2}} e^{-\frac{1}{2}(x-c)^T(A^{-1}+B^{-1})(x-c)} dx \\
&= \frac{\left|(A^{-1}+B^{-1})^{-1}\right|^{1/2}}{(2\pi)^{d/2}|A|^{1/2}|B|^{1/2}} e^{-\frac{1}{2}(a-b)^T C(a-b)} \\
&= \frac{1}{(2\pi)^{d/2}(|A||B||A^{-1}+B^{-1}|)^{1/2}} e^{-\frac{1}{2}(a-b)^T (A+B)^{-1}(a-b)} \\
&= \frac{1}{(2\pi)^{d/2}|ABA^{-1}+ABB^{-1}|^{1/2}} e^{-\frac{1}{2}(a-b)^T (A+B)^{-1}(a-b)} \\
&= \frac{1}{(2\pi)^{d/2}|ABA^{-1}+A|} e^{-\frac{1}{2}(a-b)^T (A+B)^{-1}(a-b)} \\
&= \frac{1}{(2\pi)^{d/2}|A(B+A)A^{-1}|} e^{-\frac{1}{2}(a-b)^T (A+B)^{-1}(a-b)} \\
&= \frac{1}{(2\pi)^{d/2}|A+B|^{1/2}} e^{-\frac{1}{2}(a-b)^T (A+B)^{-1}(a-b)}
\end{aligned}$$

- $\Rightarrow$ Convolution of Gaussians $G1 * G2$
  - $G_1 \sim \mathcal{N}_d(a, A), G_2 \sim \mathcal{N}_d(b, B)$

$$\begin{aligned}
G_1 * G_2(z) &= \int G_1(x) G_2(z-x) dx \\
&= \int \mathcal{N}_d(x|a, A)\mathcal{N}_d(z-x|b, B) dx \\
&= \int \mathcal{N}_d(x|a, A) \cdot \frac{1}{(2\pi)^{d/2}|B|^{1/2}} e^{-\frac{1}{2}(z-x-b)^T B^{-1}(z-x-b)} dx \\
&= \int \mathcal{N}_d(x|a, A)\mathcal{N}_d(x|z-b, B) dx \\
&= \frac{1}{(2\pi)^{d/2}|A+B|^{1/2}} e^{-\frac{1}{2}(z-(a+b))^T (A+B)^{-1}(z-(a+b))} \\
&= \boldsymbol{\mathcal{N}_d(z|a+b, A+B)}
\end{aligned}$$

1.4.5 Bayesian Interpretation of Probability

Contrasting Frequentist Estimator

- Posterior $p(\mathbf{w}|\mathcal{D}) = \dfrac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$
  - Notation
    - $\mathcal{D}$ the observed dataset
    - $\mathbf{w}$ the vector for model parameters
  - Bayesian

        - only one single dataset $\mathcal{D}$ (the observed one)
        - uncertainty expressed as distribution over $\mathbf{w}$
        - model's error: use likelihood / posterior directly (or after taking log)
        - pros
          1. naturally incorporating prior knowledge as prior distribution (of $\mathbf{w}$)
        - cons
          1. prior usually selected for mathematic convenience
    - Frequentist Estimator
        - parameters $\mathbf{w}$ already determined / fixed by 'estimator' (model)
        - error bars of the model obtained by considering the distribution over $\mathcal{D}$
        - model's error: bootstrap procedure
          1. generate dataset(s) by drawing from the observed $\mathcal{D}$ with replacement
          2. sampling L datasets with the same size as $\mathcal{D}$
          3. error = variability of predictions between the sampled datasets
        - pros
          1. protect the conclusion from false prior knowledge
        - cons
          1. sensitive to observation (extreme cases), especially under small dataset

Parameter Estimation

- Bias vs. Variance
- Taking Logarithm
    - Reason
        - monotonically increasing function $\Rightarrow \arg\max_\theta f(x;\theta) = \arg\max_\theta \log f(x;\theta)$
        - simplify mathematical analysis $\Rightarrow \prod \to \sum$
        - numerical stability $\Rightarrow$ avoid $\prod$(small probabilities)
          (may otherwise underflow the numerical precision)
- Maximum Likelihood Estimation for Gaussian
    - Notation
        - $X = \{x_1, ..., x_N\}$: observed N data points
    - Assumption
        - data points are i.i.d. (identically and independently distributed)
        - underlying distribution is Gaussian $\mathcal{N}(\mu, \sigma)$
    - Likelihood
        - $p(X|\mu,\sigma^2) = \prod_{n=1}^{N} (x_n|\mu,\sigma^2)$
        - $\Rightarrow \ln p(X|\mu,\sigma) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$
    - Solution
        - let $\frac{\partial}{\partial\mu}\ln p(X|\mu,\sigma^2) = 0 \Rightarrow \mu_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n$
        - let $\frac{\partial}{\partial\sigma^2}\ln p(X|\mu,\sigma^2) = 0 \Rightarrow \sigma^2_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{\text{ML}})$

- Analysis
  - $\mathbb{E}[\mu_{\mathrm{ML}}] = \mathbb{E}[\frac{1}{N}\sum_{n=1}^{N} x_n] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n] = \mu$
    (as $x_1, ..., x_N$ i.i.d, drawn from $\mathcal{N}(\mu, \sigma^2)$, thus $\sim \mathcal{N}(\mu, \sigma^2)$)
    $\Rightarrow$ unbiased estimation of mean
  - $$\begin{aligned}
\mathbb{E}[\sigma^2_{\mathrm{ML}}] &= \mathbb{E}[\frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{\mathrm{ML}})^2] \\
&= \mathbb{E}[\frac{1}{N}\sum_{n=1}^{N}(x_n^2 - 2\mu_{\mathrm{ML}}x_n + \mu_{\mathrm{ML}}^2)] \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n^2] - \mathbb{E}[\frac{1}{N}\sum_{n=1}^{N}2x_n\mu_{\mathrm{ML}}] + \mathbb{E}[\mu_{\mathrm{ML}}^2] \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n^2] - 2\mathbb{E}[\mu_{\mathrm{ML}}^2] + \mathbb{E}[\mu_{\mathrm{ML}}^2] \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n^2] - \frac{1}{N^2}\sum_{i,j=1}^{N}\mathbb{E}[x_i x_j] \\
&= \frac{1}{N}\sum_{n=1}^{N}(\sigma^2 + \mu^2) - \frac{1}{N^2}[N(N-1)\mu^2 + N(\sigma^2 + \mu^2)]
\end{aligned}$$
    (by 2nd moment of Gaussian $\mathbb{E}[x^2]$ and i.i.d assumption)
    $$= \left(\frac{N-1}{N}\right)\sigma^2$$
    $\Rightarrow$ biased variance **!**
    $\Rightarrow$ unbiased variance $\hat{\sigma}^2 = \frac{N}{N-1}\sigma^2_{\mathrm{ML}} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu_{\mathrm{ML}})^2$
    interpretation: $N-1$ degree of freedom,
    (as calculating σ^2 needs μ, which help pin down x_N given $x_1, ..., x_{N-1}$)

Predictive Distribution

- Probabilistic Prediction
  - Notation
    - $\mathbf{x}, \mathbf{t}$: vector of data examples and corresponding ground truth
    - $\mathbf{w}$: model parameters
    - x, t: new data example for prediction and its ground truth
  - Prediction by Model
    - $p(t|x, \mathbf{w}')$, where $\mathbf{w}'$ is the best fit parameters founded
  - Prediction by Data
    - $p(t|x, \mathbf{x}, \mathbf{t}) = \int p(t|x, \mathbf{w})p(\mathbf{w}|\mathbf{x}, \mathbf{t})d\mathbf{w}$, where $\mathbf{w}$ marginalized over its posterior

Chapter 2

Introduction

2.1 General Concern

2.1.1 Types of Learning

Supervised Learning

- Overview
  - training data comprises examples of input vectors with corresponding target vectors
- Regression
  - output one or more continuous variable
- Classification
  - assign input to one of a finite number of discrete categories

Unsupervised Learning

- Overview
  - training data consists of a set of input vectors without target vectors
- Clustering
  - Goal: discover groups of similar examples
- Density Estimation
  - Goal: determine the distribution of data within the input space
- Dimension Reduction
  - Goal: project data into low dimension, for the purpose of such as visualization

Reinforcement Learning

- Overview
  - input with time series & discover optimal output by a process of trial and error
- Goal
  - find actions to take under given circumstance to maximize a reward

2.2 Decision Theory

2.2.1

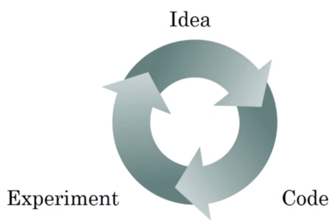
2.3 Information Theory

2.4 Recommended Practice

2.4.1 Data & Dataset

Train-Val-Test

- Reason
  - Iterative Process



    - intuition usually do NOT transfer across domains (NLP, CV, Search, etc.)
    - do NOT hope to have the correct hyperparameters at the first try

  ⇒ need feedback from experiment result
  ⇒ make sure the feedback is CORRECT and FAST

- Recommended Usage
  - Splitting
    - classic split for small dataset ⇒ train:val:test = 80 : 20 : 20, or K-fold
    - in big data (e.g. 100 million) ⇒ train:val:test = 98 : 1 : 1

    (as long as val-test sets cover enough data variance)
  - Training Set
    - data used to find the model parameter estimation (used for learning process of model)
      ⇒ over-fit by complex model
  - Validation Set (Val)
    - data to indicate generalization ability of a range of trained models
      ⇒ for model comparison, selection & hyperparameters tunning
      (feedback is correct if enough various input covered)
  - Test Set
    - evaluate the **generalization ability** of final selected & trained model (indication correct if enough various input covered)

- Special Usage
  - No Val Set
    - may use the ”test” set as val set ⇒ generalization ability NOT reported
  - Training on Train-Val Set

        - to utilize as many data as possible for ultimate performance
          ⇒ okay case of "no val set"

- Potential Problem
    - Mismatched Distribution across Sets
        - classic supervised learning assumption: all sets drawn from SAME distribution
        - other (e.g. transfer/adaptive) learning focus on violation of such assumption

      ⇒ yet, make sure val&test set from the SAME distribution as the desired one
    - **Overfitting Val Set**
        - iteratively tunning model is a processing of learning (fitting to the val set)
          ⇒ with enough iteration, val set can be overfitted
        - may consider test set as 2^{nd} val set, and further have $3^{\text{rd}}, 4^{\text{th}}$... val sets
    - Limited Data
        - better model ⇒ more training data
        - ⇒ less validation ⇒ noisy estimation of generalization ability

Train-Test

- K-fold Cross Validation
    - Procedure
        - split all data into K folds, $K-1$ folds for train, 1 for validation
        - ⇒ average over all C_K^1 combination to indicate the generalization ability
        - extreme case: leave-out-one ⇒ $K = N$, where N is number of all data
    - Cons:
        - $\mathcal{O}(K)$ ⇒ slow, especially if training process already slow
          ⇒ trade off between time vs. constraint on validation
        - hence, **not** often used in big data era

Data Augmentation

Data Preprocessing

- Mean Centering
    - Practice
        - for all training examples, compute mean (on each features) $\mu = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$,
          where $\{\mathbf{x}_1, ..., \mathbf{x}_N\} = X_{\text{train}}$ the training set
        - preprocess each $\mathbf{x} \in X_{\text{train}}, X_{\text{val}}, X_{\text{test}}$ to be $\mathbf{x}' = \mathbf{x} - \mu$ (all data go through the same process)
    - Pros
        - (training) data has a zero mean (statistically, most data close to 0)
    - Cons
        - different features may reside in various scales
- Standardizing
    - Practice

  - compute mean μ, standard deviation $\sigma = \left(\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \mu)^2\right)^{1/2}$, where $\{\mathbf{x}_1, ..., \mathbf{x}_N\} = X_{\text{train}}$ the training set
  - preprocess each $\mathbf{x} \in X_{\text{train}}, X_{\text{val}}, X_{\text{test}}$ to be $\mathbf{x}' = \frac{\mathbf{x}-\mu}{\sigma}$
    (all data go through the same process)
  - note: with big data, usually computed iteratively due to limited memory
- Pros
  - (training) data has zero mean & unit variance
    $\Rightarrow$ approximated to normal distribution
  - for deep learning: different features in same small range close to 0
    $\Rightarrow$ weights for different features are in roughly the same scale
    $\Rightarrow$ easier to train

2.4.2 Project Structuring

2.5 Model Analysis

2.5.1 Bias and Variance

Overview

- Low Bias, High Variance (Over-fitting)
  - Symptom
    - good performance on training set & poor generalization
      (good on train; bad on val)
    - $\Rightarrow$ good at fitting training set; bad at representing modeling underlying data source
  - Cause
    - too much representation ability (to fit even the noise)
    - directly model the likelihood instead of posterior
  - Remedy
    - larger dataset
    - regularization (model the posterior by accounting prior)
- High Bias, Low Variance (Under-fitting)
  - Symptom
    - bad at fitting training examples & modeling underlying data source
      (bad at train & val)
    - $\Rightarrow$ poor performance on training set & good generalization (though meaningless)
  - Cause
    - lack of representation ability (not enough flexibility)
  - Remedy
    - try model with better representative ability (more complexity, flexibility)
- High Bias, High Variance (Over&Under-fitting)
  - Symptom

      - bad at fitting some general cases; while good at some rare and special cases (especially in high dimensional space)
    - Cause
      - model probably not suitable for the dataset
    - Remedy
      - switch to other types of model
      - dataset preprocessing

- Low Bias, Low Variance
    - Behavior
      - good at fitting training examples & modeling underlying data source (good at train & val)
      - $\Rightarrow$ good performance on training set & good generalization ("good"= close to base performance, e.g. human ability in supervised learning)

Guideline

- Solving High Bias
    - Increasing Model Capability
      - increase complexity: more weights, latent variable / hidden layer etc.
      - use more suitable model specifically designed for the data (e.g. CNN for image)

  $\Rightarrow$ until fitting training set well

- Solving High Variance
    - Data Augment
      - get/simulate more training data (via crowd sourcing, distortion, GANs, etc.)
    - Model Regularization
      - control the complexity of model (e.g. L0/1/2 normalization)

- Solving Trade-off
    - Iterative Process
      - solve bias, then solve variance, iteratively
    - Complexity + Data/Regularization
      - increase complexity to solve bias without hurting variance (via more data/regularization)
      - more data/regularization to solve variance without hurting bias (with enough complexity)

L2 regularization also called "weight decay", as in gradient decent, weight is multiplied by a <1 number due to L2 term

2. Interaction with regularization:

- Improper λ: - large λ =¿ high bias - small λ =¿ high variance - Choosing λ: - try $\lambda = 0, 0.01, 0.02, 0.04, ..., 10$ - select the model with lowest $J_{cv}(\theta)$ without regularization term

3. Interaction with training set size:

- Normal Learning curve:

![Normal learning curve](../../Machine

- Learning curve with high bias:

- where getting more training data **doesn't** help
![Learning curve with high bias](../../Machine
- Learning curve with high variance:
- where getting more training data **helps**
![Learning curve with high variance](../../Machine
4. Ways to fix:
- High bias: - more features / more polunomial terms of features - decreasing λ
- High variance:
- larger data set - fewer features - increasing λ
- **In neural network:**
- High bias =¿ larger neural networks (more hidden layers / more units in one layer)
- High variance =¿ smaller neural networks
Larger network with regularization (λ) is more powerful

2.5.2 Evaluating Hypothesis

1. Data set =¿ training set (60
- randomly split
2. Cross Validation:
- estimation of the generalization error, inaccurate because:
- finite trainning set - finite corss validation set
- S-fold Cross Validation:
- devide whole set of data into S sets
- for $i \in (1, S)$
choose the i^{th} set as cross validation (or test set in some cases)
rest of the sets are called traning set
run the procedure (mentioned later) on the traning set
estimate generalization in CV set
3. Test set error
- Linear regression: $J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x^i_{test}) - y^i_{test})^2$ - Logistic regression: $J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} (y^i_{test} log h_\theta(x^i_{test}) + (1 - y^i_{test}) log h_\theta(x^i_{test}))$ - Misclassfication error: $J_{test}(\theta) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(h_\theta(x^i_{test}), y^i_{test})$
-

$$err(h_\theta(x), y) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 y = 0 \text{ or } h_\theta(x) < 0.5 y = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

4. Choosing procedure:
- Minimize traning error $J_{train}(\theta)$ - Select a model with lowest $J_{cv}(\theta)$ - Estimate generalization error as $J_{test}(\theta)$

2.5.3 Error Analysis

1. Procedure: - Algorithm (trained) misclassifies n data in cross validation set - Classify these n data and rank them - Maybe more features are found 2. Feature selection =¿ Numerical evaluation - =¿ test algorithm with / without this feature on **CV set** (compare error rate)

2.5.4 Skewed classes

1. Precision / Recall
- — — **Actual 1** — **Actual 0** — — ————— — ———— — ———— — — **Predicted 1** — True positive — False positive — — **Predicted 0** — False negative — True negative —

- **Precision** $= \frac{\text{True positive}}{\text{Predicted positive}} = \frac{\text{True positive}}{\text{True pos + False pos}}$
- **Recall** $= \frac{\text{True positive}}{\text{Actual positive}} = \frac{\text{True positive}}{\text{True pos + False neg}}$

2. Evaluation with precision/recall
- Predict 1 if $h_\theta(x) \geq \epsilon$, 0 if $h_\theta(x) < \epsilon$
- larger ϵ =¿ higher precision, lower recall (more confident) - smaller ϵ =¿ lower pecision, higher recall (avoid missing)

!\[Posiible Precision -Recall curev](../../Machine Learning/Statistical 0Machine Learning/Posiible Precision -Recall curev.png)

3. Compare precision/recall num
- $\text{F}_1 Score = 2\frac{PR}{P+R}$, P as precision, R as recall - higher better, on cross validation set

4. High precision & high recall:
- **large num of features (low bias) + large sets of data (low variance)**

2.6 Supervised Learning

- Feature normalization: $\forall x_{ij} \in X, x_{ij} = \frac{x_{ij}-\mu_j}{\sigma_j^2}$, $X : [instance][feature]$, without $[1...1]^T$ in 1st column $X = [x_1, x_2, ..., x_m]$, m instances in total
- Regularization: add penalty for θ being large into cost function
- $J(\theta) = ... + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$, **bias θ_0 shouldn't be penalized**

2.7 Linear Regression

- Notation
  - t: observed data
  - $y(\mathbf{x}, \mathbf{w}) = \sum_{i=0}^{M}\phi_i(\mathbf{x})w_i = \mathbf{w}^T\phi(\mathbf{x})$: model generating ground truth, with
    - $\mathbf{w}$: weight vector
    - $\phi(\mathbf{x})$: basis function for feature vector $\mathbf{x}$, with usually $\phi_0(\mathbf{x}) = 1$ as bias
- Assumption
  - Deterministic Model with Observation Noise
    - $t = y(x, w) + \epsilon$, where
      $\epsilon \sim \mathcal{N}(0, \beta^{-1})$ is Gaussian noise where precision (inverse variance) β
    - $\Rightarrow$ consequence
      1. likelihood $p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$
      2. $\mathbb{E}[t|\mathbf{x}] = \int t \cdot p(t|\mathbf{x})dt = y(\mathbf{x}, \mathbf{w})$
      3. unimodal distribution $p(t|\mathbf{x}) \Rightarrow$ extended by conditional mixture model
- Joint Likelihood
  - $P(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N}\mathcal{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1})$, where
    - $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}, \mathbf{t} = \{t_1, ..., t_N\}$

  - Log Likelihood
    - $\ln P(\boldsymbol{y}|X,\theta,\beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$

- Log Posterior leads to regularization
  - Maximizing the likelihood function $\Rightarrow$ (often) excessively complex models & over-fitting
  - Regularization term comes from the Prior:
    - assume Prior $p(\theta) = \mathcal{N}(\theta|0, \alpha^{-1}I)$, so that Posterior & Prior are of the same distribution
      to maximize log Posterior :
      $\Rightarrow \ln p(\theta|X) \propto -\frac{\beta}{2}\sum_{i=1}^{n}(y^i - h_\theta(x))^2 - \frac{\alpha}{2}\theta^T\theta + const$
    - If $\alpha \to 0$ (Prior is most useless), maximise Posterior is equivalent to maximizing likelihood
    - Maximize Posterior $\Leftrightarrow$ Minimize cost function with regularization, where $\lambda = \alpha/\beta$

- Predictive Distribution: $p(y|x,X,Y)$
  - $p(y|x,X,Y) = \int p(y,\theta|x,X,Y)d\theta = \int p(y|\theta,x,X,Y)p(\theta|x,X,Y)d\theta$
  - $p(y|\theta,x,X,Y) = p(y|\theta,x) = \mathcal{N}(y|h(x,\theta),\beta^{-1})$
    based on assumption: $y = y(x,\theta) + \epsilon$, where ϵ is Guassian noise
    $p(\theta|x,X,Y) = p(\theta|X,Y) =$ posterior
  - $\Rightarrow p(y|x,X,Y) = \int p(y|\theta,x)p(\theta|X,Y)d\theta$
  - Expected Lost $= (bias)^2 + variance + noise$

- Notation:
  - $t = y(x,w) + \epsilon,$ where ϵ is Gaussian noise
  - $\hat{y}$ is prediction function to approximate $y = y(x,w)$

- Procedure:
  - $\mathbb{E}[(t-\hat{y})^2] = \mathbb{E}[t^2 - 2t\hat{y} + \hat{y}^2]$
    $= \mathbb{E}[t^2] + \mathbb{E}[\hat{y}^2] - \mathbb{E}[2t\hat{y}]$
    $= \mathrm{Var}[t] + \mathbb{E}[t]^2 + \mathrm{Var}[\hat{y}] + \mathbb{E}[\hat{y}]^2 - 2y\mathbb{E}[\hat{y}]$
    $= \mathrm{Var}[t] + \mathrm{Var}[\hat{y}] + (y^2 - 2y\mathbb{E}[\hat{y}] + \mathbb{E}[\hat{y}]^2)$
    $= \mathrm{Var}[t] + \mathrm{Var}[\hat{y}] + (y - \mathbb{E}[\hat{y}])^2$
    $= \mathrm{Var}[t] + \mathrm{Var}[\hat{y}] + \mathbb{E}[t-\hat{y}]^2$
    $= \sigma^2 + \mathrm{Var}[\hat{y}] + \mathrm{Bias}[\hat{y}]^2$
    where $\sigma^2 = \mathrm{Var}[\epsilon]$ is the noise
    (formula used: $\mathrm{Var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 \Leftrightarrow \mathbb{E}[x^2] = \mathrm{Var}[x] + \mathbb{E}[x]^2$)
  - Matrix inverse can be evil & avoid inverse operation:
    $A = U\Lambda U^T$, where Λ is diagonal matirx
    $=> A^{-1} = U\Lambda^{-1}U^T$
    but number on the diagonal line of Λ can be small =¿ maybe 0 depending on accuracy of computer

2.8 Bayesian Regression

- Assumption:
  - $t = y(x, w) + \epsilon,$ where ϵ is Gaussian noise; $y(x, w)$ approximated by $\phi(x)w$
- Bayesian view:
- Gaussian Prior : $p(w) = \mathcal{N}(w|m_0, S_0)$
  Reason: to be conjugate
- Likelihood : $p(\boldsymbol{t}|w) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T\phi(x_n), \beta^{-1}) = \mathcal{N}(\boldsymbol{t}|\Phi w, \beta^{-1}I)$
- $\Rightarrow$ Posterior : $p(w|\boldsymbol{t}) = \mathcal{N}(w|m_N, S_N)$
  where $m_N = S_N(S_0^{-1}m_0 + \beta\Phi^T\boldsymbol{t}), S_N^{-1} = S_0^{-1} + \beta\Phi^T\Phi$
- Maximum Likelihood:
  - Likelihood : $p(\boldsymbol{t}|w) = \prod_{n=1}^{N} \mathcal{N}(t_n|\phi(x_n)w, \beta^{-1})$
    - meaning: how probable the observed dataset is w.r.t the model setting (under parameter w)
  - $\ln \text{Likelihood} = \sum_{n=1}^{N} [-\ln \frac{\beta}{\sqrt{2\pi}} - \frac{\beta}{2}(t_n - \phi(x)w)^2]$
  - $\frac{\partial}{\partial w} \ln \text{Likelihood} = \beta\Phi^T(\boldsymbol{t} - \Phi w)$
    let $\frac{\partial}{\partial w} \ln \text{Likelihood} = 0$
    $\Rightarrow w_{ML} = (\Phi^T\Phi)^{-1}\Phi^T\boldsymbol{t}$
  - $\frac{\partial}{\partial \beta} \ln \text{Likelihood} = -N\beta^{\frac{1}{2}} + \beta^{\frac{3}{2}}(\boldsymbol{t} - \Phi w)^T(\boldsymbol{t} - \Phi w)$
    let $\frac{\partial}{\partial \beta} \ln \text{Likelihood} = 0$
    $\Rightarrow \beta^{-1} = \frac{1}{N}(\boldsymbol{t} - \Phi w)^T(\boldsymbol{t} - \Phi w)$
    Note: solve $w = w_{ML}$ first
- Maximum Posterior:
  - $\text{Posterior} = p(w|\boldsymbol{t}), \text{Prior} = p(w), \text{Likelihood} = p(\boldsymbol{t}|w), \text{Normalization} = p(t)$
    $\Rightarrow \text{Posterior} = \frac{\text{Likelihood*Prior}}{\text{Normalization}}$
    $\Rightarrow \text{Posterior} \propto \text{Likelihood*Prior}$
  - assume Prior $p(w) = \mathcal{N}(w|m_0, S_0)$,
    so that Prior & Likelihood are conjugate $\Rightarrow$ Gaussian Posterior
  - Likelihood $p(\boldsymbol{t}|w) = \prod_{n=1}^{N} \mathcal{N}(t_n|\phi(x_n)w, \beta^{-1}) = \mathcal{N}(\boldsymbol{t}|\Phi w, \beta^{-1}I)$
  - $\Rightarrow$ Posterior $p(w|\boldsymbol{t}) = \mathcal{N}(w|m_N, S_N)$,
    where $m_N = S_N(S_0^{-1}m_0 + \beta\Phi^T\boldsymbol{t}), S_N^{-1} = S_0^{-1} + \beta\Phi^T\Phi$
    $\Rightarrow w_{MAP}$ = mean of the Gaussian = m_N
    Note: can also get w_{MAP} from taking gradient

- Simple Prior:
  Prior $p(w) = \mathcal{N}(w|0, \alpha^{-1}I)$
  $\Rightarrow$ Posterior $p(w|\boldsymbol{t}) = \mathcal{N}(w|m_N, S_N)$,
  where $m_N = \beta(\alpha I + \beta\Phi^T\Phi)\Phi^T\boldsymbol{t}, S_N^{-1} = \alpha I + \beta\Phi^T\Phi$
  $w_{MAP} \to w_{ML},$ when $\alpha \to 0$ (most useless Prior)
- Maximize Posterior $\Leftrightarrow$ Minimize cost function with regularization:
  Simple Prior $\Rightarrow \ln p(w|\boldsymbol{t}) = -\dfrac{\beta}{2}(\boldsymbol{t} - \Phi w)^T(\boldsymbol{t} - \Phi w) - \dfrac{\alpha}{2}w^T w + const$
- If $\alpha \to 0$ (Prior is most useless), maximize Posterior is equivalent to maximizing likelihood
- Maximize Posterior equal to minimize sum-of-squares error function with the addition of a quadratic regularization term with $\lambda = \alpha/\beta$
- Regularization term comes from the Prior
- Predictive Distribution:
- Assume: Prior : $p(x|\alpha) = \mathcal{N}(x|0, \alpha^{-1}I)$, $(m_0 = 0, S_0 = \alpha^{-1}I)$
- $p(t|x, X, \boldsymbol{t}) = \displaystyle\int p(t|w, x)p(w|X, \boldsymbol{t})dw$
- $\Rightarrow p(t|x, X, \boldsymbol{t}) = \mathcal{N}(t|m_N^T\phi(x), \sigma_N^2(x))$
  where $\sigma_N^2(x) = \frac{1}{\beta} + \phi(x)^T S_N \phi(x); m_N, S_N$ from Posterior($m_N = w_{MAP}$)
- Sequential data:
  - Posterior from previous data $\Leftrightarrow$ the Prior for the arriving data
  - Sequential data and data in one go is equivalent when finfding the Porsterior
- Gradient descent
  - Hypothesis function:
    - $h_\theta(x) = x\theta$, $\theta = [\theta_0, \theta_1, ..., \theta_n]^T$, $x = [x_0, x_1, ..., x_n], x_0 = 1$
  - x is one instance
  - Cost function: $J(\theta) = \dfrac{1}{2m}\displaystyle\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 + \dfrac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$
  - Update rule: $\forall\theta_j \in \theta, \theta_j := \theta_j - \alpha\frac{\partial J(\theta)}{\partial\theta_j}$, $\dfrac{\partial J(\theta)}{\partial\theta_j} = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}[(h_\theta(x^i) - y^i)x_j^i] + \dfrac{\lambda}{m}\theta_j$ - $\dfrac{d}{d\theta}J(\theta) = \dfrac{1}{m}((X\theta - y)^T X)^T + \dfrac{\lambda}{m}[0, \theta_1, ..., \theta_m]^T$ (θ_0 shouldn't be penalized)
  - simultaneously for all $\theta_j \in \theta$
  - Normal equation (mathematical solution)
    - $\theta = (X^T X)^{-1}X^T y$

2.9 Logistic Regression (Classification)

- Decision Theory:

- classes $C_1, ..., C_K$, decision regions $\mathcal{R}_1, ..., \mathcal{R}_K$ - Minimze $p(mistake) = \sum_{k=1}^{K}(\int_{\mathcal{R}_k} \sum_{i\neq k} p(x, C_i)dx)$ (can have weight on each misclassfication though) - Maximize $p(correct) = \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(x, C_k)dx$

- Models for Decision Problems:

- Find a discriminant function - Discriminative Models: less powerful, yet less parameter =¿ easier to learn - Infer **posterior** $p(C_k|x)$, C_k: $x \in C_k$, x is examples in trainning set - Use decision theory to assign a new x - Generative Models: more powerful, but computationally expensive - Infer conditional probabilities $p(x|C_k)$ - Infer prior $p(C_k)$ - Find **either** **posterior** $p(C_k|x)$, **or** **joint distribution** $p(x, C_k)$ (using Bayes' theorem) - Use decision theory to assign a new x - **=¿ Able to create synthetic data using $p(x)$**

- Naive Bayes on Discrete Features:

- Assumption:

- Discrete Features: data point $x \in \{0, 1\}^D$

- Naive Bayes: all features conditioned on class C_k are independent with each other

$\Rightarrow p(x|C_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i}(1 - \mu_{ki})^{1-x_i}$

1. Linear Discriminant (Least Squares Approach)

- Prediction:

- $y(x) = xw + w_0$, with bias $= w_0$, where $w = [w_1, ..., w_n]^T, x = [x_1, ..., x_n]$ - $y(x) > 0$: positive confidence to assign x to current class - $-w_0$ called threshold sometimes

- Decision Boundary $y(x) = w^T x + w_0 = 0$:

- w is orthogonal to vectors on the boundary:

assume x_1, x_2 on the boundary

$\Rightarrow 0 = y(x_1) - y_(x_2) = (x_1 - x_2)w$

- Distance from origin to boundary is $-\frac{w_0}{\|w\|}$:

assume distance is k

$\Rightarrow k\frac{w}{\|w\|}$on boundary, thus $k\frac{w}{\|w\|}w + w_0 = 0$

$\Rightarrow k = -\frac{w_0}{\|w\|}$

- $y(x)$ is a signed measure of distance from point x to boundary:

assume distance is r

$$\Rightarrow y(x) = \overbrace{(x_\perp + r\frac{w}{\|w\|})}^{x} w + w_0 = \overbrace{x_\perp w + w_0}^{0} + r\|w\| = r\|w\|$$

$\Rightarrow r = \frac{y(x)}{\|w\|}$

- Multi-class (k-classes):

- prediction: x is of class $\mathcal{C}_k$ if $\forall j \neq k, y_k(x) > y_j(x)$

$\Rightarrow y(x) = xW$, where $W = [w_1, ..., w_k], \forall x_i \in X, x_{i0} = 1$ (bias), $y(x)$ is 1-of-k coding

- sum-of-squares error: $E_D(W) = \frac{1}{2}\text{tr}\{(XW - T)(XW - T)^T\}$

$\Rightarrow$ optimal $W = (X^T X)^{-1} X^T T$

note that $tr\{AB\} = A^T B^T$

2. Fisher's Linear Discriminant

- Basic idea:

- Take linear classification $y = w^T x$ as dimensionality reduction (projection onto 1-D) - =¿ find a projection (denoted by vector w) which maximally preserves the class separation - =¿ if $y > -w_0$ then class C_1, otherwise C_2

- Goal:

- Distance within one class is small - Distance between classes is large

- Mean & Variance of Projected Data:
- Mean: $\widetilde{m}_k = w^T m_k$, where $m_k = \frac{1}{N_k}\sum_{x \in C_k} x$ - Variance: $\widetilde{s}_k = \sum_{x \in C_k}(w^T x - w^T m_k)^2 = w^T[\sum_{x \in C_k}(x - m_k)(x - m_k)^T]w$
- 2-Classes to 1-D line:
- Maximize Fisher criterion: $J(w) = \frac{|\widetilde{m}_1 - \widetilde{m}_2|^2}{\widetilde{s}_1^2 + \widetilde{s}_2^2}$
- Between-class covariance: $S_B = (m_1 - m_2)(m_1 - m_2)^T$
- Within-class covariance: $S_k = \sum_{n \in C_k}(x_n - m_k)(x_n - m_k)^T$

$\Rightarrow J(w) = \frac{w^T S_B w}{w^T S_W w}$
- Lagrangian: $L(w, \lambda) = w^T S_B w + \lambda(1 - w^T S_W w)$
fix $w^T S_W w$ to 1 to avoid infinite solution (any multiple of a solution is a solution)
$\Rightarrow \frac{\partial}{\partial w} L = 2S_B w - 2\lambda S_W w = 0$
$\Rightarrow S_B w = \lambda S_W w$
$\Rightarrow ({S_W}^{-1} S_B)w = \lambda w$
To maximize $J(w)$, w is the largest eigenvector of ${S_W}^{-1} S_B$ if S_W invertible
- K-classes to a d-D subspace: N_k is num in class k, N is the total example num
- Between-class covariance: $S_B = \sum_{k=1}^{K} N_k(m_k - m)(m_k - m)^T$, where $m = \frac{1}{N}\sum_{n=1}^{N} x_n$
reduce to $(m_1 - m_2)(m_1 - m_2)^T$ when K=2 (constant ignored)
- Within-class covariance: $S_W = \sum_{k=1}^{K} S_k$, where $S_k = \sum_{n \in C_k}(x_n - m_k)(x_n - m_k)^T$, $m_k = \frac{1}{N_k}\sum_{n \in C_k} x_n$
- Maximize Fisher criterion: $J(w) = \frac{tr\{W^T S_B W\}}{tr\{W^T S_W W\}}$
- Lagrangian:
Solve for each $w_i \in W \Rightarrow ({S_W}^{-1} S_B)w_i = \lambda_i w_i$
$\Rightarrow W$ conosists of the largest d eigenvectors
${S_W}^{-1} S_B$ is not guaranteed to be symmetric $\Rightarrow W$ might not be orthogonal
Need to minimize the whole covariance matrix ($J(w)$ as a matrix) $\Rightarrow$ not choosing same eigenvectors twice
- Maximum Possible Projection Directions $= K - 1$:
$r({S_W}^{-1} S_B) \leq \min(r({S_W}^{-1}), r(S_B)) \leq r(S_B)$
$r(S_B) \leq \sum_K r((m_k - m)(m_k - m)^T) = K$, as $r(m_k - m) = 1$
$\sum_K m_k = m \Rightarrow r(m_1 - m, ..., m_K - m) = K - 1$
$\Rightarrow r(S_B) \leq K - 1$
$\Rightarrow r({S_W}^{-1} S_B) \leq K - 1$

3. Perceptron Algorithm
- Generalised linear model $y = f(w^T \phi(x))$, where $\phi(x)$ is basis function; $\phi_0(x) = 1$
- Nonlinear activation funtion: $f(a) = \begin{cases} 1, & a \geq 0 \\ -1, & a < 0 \end{cases}$
- Target coding: $t = \begin{cases} 1, & \text{if } C_1 \\ -1, & \text{if } C_2 \end{cases}$
- Cost function:

- All correctly classified patterns: $w^T\phi(x_n)t_n > 0$
- Add the errors for all misclassified patterns (denoted as set $\mathcal{M}$):

$\Rightarrow E_P(w) = -\sum_{n\in\mathcal{M}} w^T\phi(x_n)t_n$

- Algorithm: (Aim: minimize total num of misclassified patterns)
- loop

choose a traning pair (x_n, t_n)

update the weight vector w: $w = w - \eta\nabla E_p(w) = w + \phi_n t_n$

where η=1 because $y = f(\cdot)$ does not depend on $\|w\|$

- Perceptron Convergence Theorem:
- If the training set is linearly separable, the perceptron algorithm is guaranteed to find a solution in a finite number of steps

(Also is the algorithm to find whether the set is linearly separable =¿ Halting Problem)

4. Maximum Likelihood

- Assumption: - $p(x|C_k) \sim \mathcal{N}(\mu_k, \Sigma)$, and all $p(x|C_k)$ share the same Σ - $p(C_1) = \pi, p(C_2) = 1-\pi$, π unknown - Likelihood of whole data set $\boldsymbol{X}, \boldsymbol{t}$, N is the num of data - $p(\boldsymbol{X}, \boldsymbol{t}|\pi, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N}[\pi\mathcal{N}(x_n|\mu_1, \Sigma)]^{t_n}[(1-\pi)\mathcal{N}(x_n|\mu_2, \Sigma)^{1-t_n}]$ $\rightarrow$ when info of label t lost: mixture of Gaussian - $\ln(\text{Likelihood}) = \sum_{n=1}^{N}[t_n(\ln\pi + \ln\mathcal{N}(x_n|\mu_1, \Sigma)) + (1-t_n)(\ln(1-\pi) + \ln\mathcal{N}(x_n|\mu_2, \Sigma))]$ - Parameters when maximum reached: - $\pi = \frac{N_1}{N_1+N_2}$, N_1 is the num of class C_1 - $\mu_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n x_n, \mu_2 = \frac{1}{N_2}\sum_{n=1}^{N}(1-t_n)x_n$, (mean of each class) - $\Sigma = \frac{N_1}{N}S_1 + \frac{N_2}{N}S_2$, where $S_k = \frac{1}{N_k}\sum_{n\in C_k}(x_n-\mu_k)(x_n-\mu_k)^T$

5. Logistic Regression

- Sigmoid function: $\sigma(a) = \frac{1}{1+e^{-a}}$
- $p(x|C_k) \sim \mathcal{N} \implies p(C_k|x) = \sigma(w^T x + w_0)$ (2-classes) (Generative model)
- Assumption:

$p(x|C_k) = \mathcal{N}(\mu_k, \Sigma)$ (can also be a number of other distributions)

$\forall k, p(x|C_k)$ shares the same Σ

-

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} = \sigma(a),$$

$$\begin{aligned}
\text{where } a &= \ln\frac{p(x, C_1)}{p(x, C_2)} \\
&= \ln\frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)} \\
&= \ldots \text{(assumption applied)} \\
&= \ln\frac{\exp(\mu_1^T\Sigma^{-1}x - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1)}{\exp(\mu_2^T\Sigma^{-1}x - \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2)} + \ln\frac{p(C_1)}{p(C_2)} \\
\implies a &= w^T x + w_0 \text{ where,} \\
w &= \Sigma^{-1}(\mu_1 - \mu_2) \\
w_0 &= -\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 + \ln\frac{p(C_1)}{p(C_2)}
\end{aligned}$$

- $\implies p(C_1|x) = \sigma(w^T x + w_0)$

$\Rightarrow$ Find parameters in Gaussian model using Maximal Likelihood Sulotion

as: $p(C_1|x) \propto p(x|C_1)p(C_1) = p(x, C_1)$
- Generalize to K-classes:
$a_k(x) = \ln[p(x|C_k)p(C_k)], p(C_k|x) = \dfrac{\exp(a_k)}{\sum_i \exp(a_i)}$
$\Rightarrow a_k(x) = w_k^T x + w_{k0}, \text{where } w_k = \Sigma^{-1}\mu_k; w_{k0} = -\dfrac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + p(C_k)$
- Assume directly $p(C_k|x) = \sigma(w^T x + w_0)$ (2-classes) (Discriminative model)
- Assume directly: $p(C_1|w, x) = \sigma(w^T x), x_0 = 1$
$\Rightarrow$ less parameters to fit (compared to Gaussian)
- Likelihood function:
$p(\boldsymbol{t}|w, X) = \prod_{n=1}^{N} p_n^{t_n}(1 - p_n)^{1-t_n}$, where, $p_n = p(C_1|x_n), t_n$ is the class of x_n
Define error function :
$E(w) = -\ln(Likelihood) = -\sum_{n=1}^{N}[t_n \ln p_n + (1 - t_n)\ln(1 - p_n)]$
$\Rightarrow \nabla E(w) = \sum_{n=1}^{N}(p_n - t_n)x_n$
- Find Posterior $p(w|\boldsymbol{t})$:
Likelihood is product of sigmoid
Conjugate Prior for "sigmoid distribution" is unknown
$\Rightarrow$ Assume Prior $p(w) = \mathcal{N}(w|m_0, S_0)$
$\Rightarrow \ln p(w|\boldsymbol{t}) \propto -\dfrac{1}{2}(w - m_0)^T S_0^{-1}(w - m_0) + \sum_{n=1}^{N}[t_n \ln p_n + (1 - t_n)\ln(1 - p_n)]$
find w_{MAP}, calculate $S_N = -\nabla\nabla \ln p(w|\boldsymbol{t}) = S_o^{-1} + \sum_{n=1}^{N} p_n(1 - p_n)\phi_n\phi_n^T$
$\Rightarrow p(w|\boldsymbol{t}) \simeq \mathcal{N}(w|w_{MAP}, S_N)$, via Laplace Approximation
- Laplace Approximation:
- Fit a guassian to $p(z)$ at its **mode** (mode of $p(z)$: point where $p'(z) = 0$)
- Assume $p(z) = \frac{1}{Z}f(z)$, with normalization $Z = \int f(z)dz$
Taylor expansion of $\ln f(z)$ at z_0: $\ln f(z) \simeq \ln f(z_0) - \frac{1}{2}A(z - z_0)^2$,
where $f'(z_0) = 0, A = -\frac{d^2}{dz^2}\ln f(z)|_{z=z_0}$
Take its exponentiating: $f(z) \simeq f(z_0)\exp{-\frac{A}{2}(z - z_0)^2}$
$\Rightarrow$ Laplace Approximation $= (\dfrac{A}{2\pi})^{1/2}\exp{-\dfrac{A}{2}(z - z_0)^2}$, where $A = \dfrac{1}{\sigma^2}$
- Requirement:
$f(z)$ differentiable to find a critical point
$f''(z_0) < 0$ to have a maximum & so that $\nabla\nabla \ln f(z_0) = A > 0$ as $A = \frac{1}{\sigma^2}$
- In Vector Space: approximate $p(z)$ for $z \in \mathcal{R}^M$
Assume $p(z) = \frac{1}{Z}f(z)$
Taylor expansion: $\ln f(z) \simeq \ln f(z_0) - \frac{1}{2}(z - z_0)^T A(z - z_0)$,
Hessian $A = -\nabla\nabla \ln f(z)|_{z=z_0}$

$$\Rightarrow \text{Laplace approximation} = \frac{|A|^{1/2}}{(2\pi^{M/2})}\exp{-\frac{1}{2}(z - z_0)^T A(z - z_0)} \tag{2.2}$$
$$= \mathcal{N}(z|z_0, A^{-1}) \tag{2.3}$$

- Gradient descent:
- Hypothesis function: $h_\theta(x) = \sigma(x\theta) = \frac{1}{1+e^{-x\theta}}$
- Cost function:
$J(\theta) = \dfrac{1}{m}\sum_{i=1}^{m}[-y^i \ln(h_\theta(x^i)) - (1 - y^i)\ln(1 - h_\theta(x^i))] + \dfrac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

- Update rule: $\forall\theta_j \in \theta, \theta_j := \theta_j - \alpha\frac{\partial J(\theta)}{\partial\theta_j}$, $\dfrac{\partial J(\theta)}{\partial\theta_j} = \dfrac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^i) - y^i)x_j^i] + \dfrac{\lambda}{m}\theta_j$

2.10 Latent Variable Analysis

2.10.1 Principal Component Analysis (PCA)

1. Motivation:
- Data compression (reduce highly related features) - Data visualization
2. Assumption:
- Gaussian distributions for both the latent and observed variables
3. Two Equivalent Definition of PCA:
- Linear projection of data onto lower dimensional linear space (principal subspace) such that:
$\Rightarrow$ variance of projected data is maximized
$\Rightarrow$ distortion error from projection is minimized
4. Maximum Variance Formulation
- Goal:
- project data from D dimension to M while maximizing the variance of projected data
- Eigenvalues λ of covariance matrix S express the variance of data set X in direction of corresponding eigenvectors
- Projection Vectors:
- $U = (u_1, ..., u_M)$, where $\forall i \in \{1, .., M\}, u_i \in \mathbb{R}^D$ s.t. $u_i^T u_i = 1$(only consider direction)
- Projected Data:
- Mean $= \bar{x}^T U$, where $\bar{x} = \dfrac{1}{N}\sum_{i=1}^{N} x^i$ - Variance $= tr\{U^T S U\}$, where $S = \sum_{i=1}^{N}(x^i - \bar{x})(x^i - \bar{x})^T$ (outer product)
- Lagrangian to maximize Variance:
- $L(U, \lambda) = tr\{U^T S U\} + tr\{(I - U^T U)\lambda\}$
constraint $u_i^T u_i = 1$ to prevent $u_i \to +\infty$

$$\text{For each } u_i \in U,\ \frac{\partial}{\partial u_i}L = 2Su_i - 2\lambda_i u_i = 0 \tag{2.4}$$

$$\Rightarrow Su_i = \lambda_i u_i \tag{2.5}$$

$$\Rightarrow U \text{ consists of eigenvectors corresponding to the first M large eigenvalue of } S \tag{2.6}$$

(S symmetric $\Rightarrow U$ orthogonal)
5. Minimum Error Formulation:
- Introduce Orthogonal Basis Vector for D dimension:
- $U = (u_1, ..., u_D)$
- Data representation:
- Original: $x^n = \sum_{i=1}^{D}\alpha_i^n u_i$ - Projected: $\widetilde{x^n} = \sum_{i=1}^{M} z_i^n u_i + \sum_{i=M+1}^{D} b_i u_i$
$(z_1^n, ..., z_M^n)$ is different for different x^n, $(b_{M+1}, ..., b_D)$ is the same for all x^n
- Cost function: $J = \dfrac{1}{N}\sum_{n=1}^{N}\|x^n - \widetilde{x^n}\|^2$, where $\widetilde{x^n} = \sum_{i=1}^{M} z_i^n u_i + \sum_{i=M+1}^{D} b_i u_i$
- Let $\begin{cases} \dfrac{\partial}{\partial z_j^n}J = 0 \\ \dfrac{\partial}{\partial b_j}J = 0 \end{cases} \Rightarrow \begin{cases} \dfrac{1}{N}2(x^n - \widetilde{x^n})^T(-u_j) = \dfrac{2}{N}(z_j - (x^n)^T u_j) = 0 \\ \dfrac{1}{N}\sum_{n=1}^{N}2(x^n - \widetilde{x^n})^T(-u_j) = \dfrac{2}{N}\sum_{n=1}^{N}(b_j - (x^n)^T u_j) = 0 \end{cases}$

$$\Rightarrow \begin{cases} z_j = (x^n)^T u_j & j \in \{1, ..., M\} \\ b_j = \overline{x}^T u_j & j \in \{M+1, ..., D\} \end{cases}$$

Noticing $(x^n)^T u_j = (\sum_{i=1}^{D} \alpha_i^n u_i^T) u_j = a_j \Rightarrow a_j = (x^n)^T u_j$

$$\Rightarrow x^n - \widetilde{x^n} = \sum_{i=M+1}^{D} [(x^n - \overline{x})^T u_i] u_i$$

-

$$\Rightarrow J = \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{i=M+1}^{D} [(x^n - \overline{x})^T u_i] u_i \right)^T \left(\sum_{i=M+1}^{D} [(x^n - \overline{x})^T u_i] u_i \right) \tag{2.7}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{i=M+1}^{D} u_i^T ((x^n - \overline{x})^T u_i) \right) \left(\sum_{i=M+1}^{D} ((x^n - \overline{x})^T u_i) u_i \right) \tag{2.8}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} u_i^T (x^n - \overline{x})^T u_i u_i^T (x^n - \overline{x}) u_i \qquad u_i \text{ orthogonal to each other} \tag{2.9}$$

$$= \sum_{i=M+1}^{D} u_i^T \left(\frac{1}{N} \sum_{n=1}^{N} (x^n - \overline{x})^T (x^n - \overline{x}) \right) u_i \qquad \|u_i\| = 1 \tag{2.10}$$

$$\tag{2.11}$$

$$\Rightarrow J = \sum_{i=M+1}^{D} u_i^T S u_i, \text{ where } S = \frac{1}{N} \sum_{n=1}^{N} (x^n - \overline{x})^T (x^n - \overline{x})$$

- Lagrangian to Minimize J:

- $L(u_{M+1}, ..., u_D, \lambda_{M+1}, ..., \lambda_D) = \sum_{i=M+1}^{D} u_i^T S u_i + \sum_{i=M_1}^{D} \lambda_i (1 - u_i^T u_i)$

constraint $\|u_i\| = 1$ to prevent $u_i = 0$

For each u_i, $\frac{\partial}{\partial u_i} L = 2Su_i - 2\lambda_i u_i = 0$

$\Rightarrow Su_i = \lambda_i u_i$

$\Rightarrow$ To minmize J, take eigenvectors with the first $(D - M)$ small eigenvalue orthogonal to (out of) subspace

$\Leftrightarrow$ define subspace with eigenvectors with the first M large eigenvalue

-

$$\text{Intuition: } \widetilde{x_n} = \sum_{i=1}^{M} ((x^n)^T u_i) u_i + \sum_{i=M+!}^{D} (\overline{x}^T u_i) u_i \tag{2.12}$$

$$= \overline{x} + \sum_{i=1}^{M} [(x^n - \overline{x})^T u_i] u_i \tag{2.13}$$

1. Singular Value Decomposition - SVD:

- Intorduce matrix $A_{m \times n}$

- $(A^T A)_{n \times n}$ symmetric matrix (actually, Gram matrix $\rightarrow$ semi-definite) -

eigenvalue decomposition: (2.14)

$A^T A = VDV^T$, V is normalized ($v_i^T v_i = 1$) with column as eigenvector (2.15)

- $AV = (Av_1, ..., Av_n)_{m \times n}$

- Let $r(A) = r$

$$\Rightarrow r(A^T A) = r(A) = r \tag{2.16}$$

$$r(AV) = \min\{r(A), r(V)\} = \min\{r, n\} = r \tag{2.17}$$

- Reduce AV to basis $(Av_1, ..., Av_r)$
- Let $U = (u_1, ..., u_r) = (\dfrac{Av_1}{\sqrt{\lambda_1}}, ..., \dfrac{Av_r}{\sqrt{\lambda_r}})$, λ_i is i-thh eigenvalue of $A^T A$
- Orthogonal: $\forall i \neq j, u_i^T u_j = \frac{1}{\sqrt{\lambda_i \lambda_j}} v_i^T A^T A v_j = \frac{\lambda_j}{\sqrt{\lambda_i \lambda_j}} v_i^T v_j = 0$
- Unit: $\|u_i\| = \frac{\|Av_i\|}{\sqrt{\lambda_i}} = \frac{\sqrt{<Av_i, Av_i>}}{\sqrt{\lambda_i}} = 1$

$\Rightarrow U$ is standard orthogonal (orthonormal) basis
- $AV = U\Sigma,$ where $\Sigma = D^{\frac{1}{2}}$
- Expand U to orthonormal in $\mathbb{R}^m : (u_i, ..., u_m)$
- Epand corresponding part in Σ with 0
- $A = U\Sigma V^T,$ with singular value in Σ in decreasing order

2. SVD with PCA:
- X is data matrix in row (centered - zero mean)
- Eigenvectors of corvariance matrix $S = X^T X$ are in $V,$ where $X = U\Sigma V^T$
- When using $S = U\Sigma V^T \Rightarrow U = V \wedge S = V\Sigma V^T$

reduced to eigenvalue decomposition
- $S = VDV^T$ with V orthonormal:

Eigenvalues λ of covariance matrix S express the variance of data set X in direction of corresponding eigenvectors
- Projection:
- $\widetilde{X} = XV_M,$ where V_M contains first M-large eigenvectors - Projection direction is **not** unique

3. Reconstruction (approximate):
- Data is projected onto k dimension using SVD with $S = U\Sigma V^T$ - $x_{approx} = U_{reduce} \cdot z,$ U_{reduce} is n*k matrix, z is k*1 vector - ![Reconsturction from data Compression](../../Machine

4. Choosing k (num of principal components):
- choose the **smallest** k making $\dfrac{J}{V} \leq 0.01$ =¿ 99
- $[U, S, V] = \text{svd(Sigma)}$ =¿ $\dfrac{J}{V} = 1 - \dfrac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$, S is diagonal matrix

=¿ check $\frac{J}{V}$ before compress data

5. Data Preprocessing:
- PCA $vs.$ Normalization: - Normalization: Individually normalized but still correlated - PCA: create decorrelated data $-$ whitening - Whitening: projection with normalization - $S = VDV^T$, where S is Gram matrix over X^T - $\forall n, y_n = D^{-\frac{1}{2}} V^T (x^n - \overline{x})$, where $\overline{x}$ is the mean of X

$$\Rightarrow y^n \text{ has zero mean} \tag{2.18}$$

$$cov(\{y^n\}) = \frac{1}{N} \sum_{n=1}^{N} y_n y_n^T = D^{\frac{-1}{2}} V^T S V D^{\frac{-1}{2}} = I \tag{2.19}$$

6. Tips for PCA:
- Do NOT use PCA to prevent overfitting, use regularization instead - Try original data before implement PCA - Train PCA only on trainning set

2.10.2 Independent Component Analysis (ICA)

1. Goal: - Recover original signals from a mixed observed data - Source signal $S \in \mathbb{R}^{N \times K}$; mixing matrix A; Observed data $X = SA$ - Maximizes statistical independence - Find A^{-1} to maximizes independence of columns of S 2. Assumption: - At most one signal is Gaussian distributed - Ignorde amplitude and order of recovered signals - Have at least as many observed mixtures as signals - A invertible 3. Independence $vs.$ Uncorrelatedness - Independence $\Rightarrow$ Uncorrelatedness - $p(x_1, x_2) = p(x_1)p(x_2) \Rightarrow \mathbb{E}(x_1 x_2) - \mathbb{E}(x_1)\mathbb{E}(x_2) = 0$ 4. Central Limit Theorem 5. FastICA algorithm

2.10.3 t-SNE

1. Problem & Focus 2. Compared to PCA: - No whitening function to use for new data - PCA can only capture linear structure inside the data - t-SNE preserves the ¡u¿local distances¡/u¿ in the original data

2.10.4 Anomaly Detection

1. Problem to solve:

- Given dataset $x^1, x^2, ..., x^m$, build density estimation model $p(x)$ - $p(x^{test} < \epsilon) =$¿ x^{test} anomaly

2. Hypothesis function:

- $p(x) = \prod_{i=1}^{n} p(x_i), x \in R^n, \forall i \in [1, n], x_i \sim N(\mu_i, \sigma_i^2)$ - $\mu = \frac{1}{m}\sum_{i=1}^{m} x^i, \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^i - \mu)^2$ - assume $x_1, ..., x_n$ independent from each other

3. Multivariate Gaussian:

- $p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$,

$x \in R^n, \mu \in R^n, \Sigma \in R^{n\times n}$, where Σ is covariance matrix

- $\mu = \frac{1}{m}\sum_{i=1}^{m} x^i, \Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^i - \mu)(x^i - \mu)^T$ - $x_1, ...x_n$ can be correlated but **not** linearly dependent - need $m > n$ ($m \geq 10nsuggested$) or elas Σ non-invertible

4. Algorithm:

- choose features - compute μ, σ - compute $p(x)$ for new example, anomaly if $p(x) < \epsilon$

5. Evaluation (real-number):

- Labeled data into normal/anomalous set

(okay if some anomalies slip into normal set)

- training set: unlabeled data from normal set (60- CV set: labeled data from normal (20- test set: labeled data from normal (20

- Use evaluation metrics (skewed data)

6. When to use:

- Anomaly detection: - Very small num of positive data (0-20 commonly); Large num of negative data - Difficult to learn from positive data (not enough data, too many features...) - Future anomalies may look nothing like given data - Supervised Learning: - Larger num of positive & negative data - Enough positive data for algorithm to learn - Future positive example is likely to be similar to given data

7. Example:

- Anomaly detection: - Fraud detection, Manufacturing, Monitoring machines in data center... - Supervised learning: - Email spam classification (enough data), Weather prediction (sunny/rainy/etc), Cancer classification...

8. Tips:

- Non-guassian feature: transformation / using other distribution - Choosing features: compare anomaly data with normal data

2.10.5 Recommender System

1. Problem Formulation:

- $r_{i,j} = 1$ if item i is rated by user j
- $y_{i,j}$ = rating of item i given by user j
- θ^j = parameter vector for user j
- x^i = feature vector for movie i

=¿ for user j, movie i, ($r_{i,j} = 0$), predict rating $x^i\theta^j$

2. Content Based Recommendations:

- Treat each user as a seperate linear regression problem with the feature vectors of its rated items as traning set

Assume features for each items (x^i) are available and known

=¿ given X estimate Θ

- Cost Function for θ_j:

$$J(\theta^j) = \frac{1}{2}\sum_{i:r_{i,j}=1}(x^i\theta^j - y_{i,j})^2 + \frac{\lambda}{2}\sum_{k=1}^{n}(\theta^j_k)^2, \theta^j \in R^{n+1}(\theta_0 \text{ not regularized})$$

- Cost Function for Θ:

$$J(\Theta) = \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r_{i,j}=1}(x^i\theta^j - y_{i,j})^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta^j_k)^2,$$

$\theta^j \in R^{n+1}(\theta_0 \text{ not regularized}), n_u$ is num of users

- Update Rule: $\forall\theta^j_k \in \theta^j, \theta^j_k := \theta^j_k - \alpha\frac{\partial J(\Theta)}{\partial\theta^j_k}$, $\dfrac{\partial J(\Theta)}{\partial\theta^j_k} = \sum_{i:r_{i,j}=1}(x^i\theta^j - y_{i,j})x^i_k + \lambda\theta^j_k$, for $k \neq 0$ $(\theta^j \in R^{n+1})$

3. Collaborative Filtering

- **Assume preference of each users (θ^j) are available and known**

=¿ given Θ estimate X

- Cost Function for x^i: $J(x^i) = \frac{1}{2}\sum_{j:r_{i,j}=1}(x^i\theta^j - y_{i,j})^2 + \frac{\lambda}{2}\sum_{k=1}^{n}(x^i_k)^2$ - Cost Function for X:

$$J(X) = \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r_{i,j}=1}(x^i\theta^j - y_{i,j})^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x^i_k)^2$$

$x^j \in R^{n+1}(x_0 \text{ not regularized}), n_m$ is num of items - Update Rule: $\forall x^i_k \in x^i, x^i_k := x^i_k - \alpha\frac{\partial J(X)}{\partial x^i_k}$,

$$\frac{\partial J(X)}{\partial x^i_k} = \sum_{j:r_{i,j}=1}(\theta^j x^i - y_{i,j})\theta^j_k + \lambda x^i_k, \text{for } k \neq 0\ (x^i \in R^{n+1})$$

- Basic Idea:
- Randomly initialize Θ
- loop:

Estimate X

Estimate Θ

- Cost Function:

$$J(X,\Theta) = \frac{1}{2}\sum_{(i,j):r_{i,j}=1}(x^i\theta^j - y_{i,j})^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x^i_k)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta^j_k)^2, x \in R^n, \theta \in R^n$$

(the sum term in $J(\Theta)$, $J(X)$, and $J(X,\Theta)$ is the same)

- Update Rule:

- $\forall x^i_k \in x^i, x^i_k := x^i_k - \alpha\frac{\partial J(X,\Theta)}{\partial x^i_k}$, $\dfrac{\partial J(X,\Theta)}{\partial x^i_k} = \dfrac{\partial J(X)}{\partial x^i_k} = \sum_{j:r_{i,j}=1}(\theta^j x^i - y_{i,j})\theta^j_k + \lambda x^i_k, x^i \in R^n$

- $\forall\theta^j_k \in \theta^j, \theta^j_k := \theta^j_k - \alpha\frac{\partial J(X,\Theta)}{\partial\theta^j_k}$, $\dfrac{\partial J(X,\Theta)}{\partial\theta^j_k} = \dfrac{\partial J(\Theta)}{\partial\theta^j_k} = \sum_{i:r_{i,j}=1}(\theta^j x^i - y_{i,j})x^i_k + \lambda\theta^j_k, \theta^j \in R^n$

- **Algorithm**
- Initialize X, Θ to **small random values**

=¿ for symmetry breaking (similar to random initialization in neural network)

=¿ so that algorithm learns features $x^1, ..., x^{n_m}$ that are different from each other

- Minimize $J(X,\Theta)$
- Predict $y_{i,j} = x^i\theta^j$ $(Y = X\Theta)$
- Finding Related Item to Recommend
- $||x^i - x^j||$ is samll =¿ item i and j is similar
- Mean Normalization:
- Problem: if user j hasn't rated any movie, $\theta^j = [0, ..., 0]$

=¿ predicted rating of user j on all item = 0

=¿ useless prediction
- Algorithm (row version):
compute vector $\mu, \forall \mu_i \in \mu, \mu_i =$ mean of Y_i, where Y_i is the i^{th} row in Y
manipulate Y: $\forall y_{i,j} \in Y \wedge r_{i,j} = 1, y_{i,j} - = \mu_i$ =¿ the mean of each row in Y is 0
predict rating for user j on item $i = x^i \theta^j + \mu_i$
- For item i with no rating
=¿ apply column version of mean normalization
(but user with no rating is generally more important)

2.11 Large Scale Machine Learning

2.11.1 Gradient Descent with Large Dataset

1. Stochastic Gradient Descent - Problem in Big Data: - Updating θ becomes computationally expensive in batch gradient decent
- Cost Function: - Cost function on single data: $cost(\theta, (x^i, y^i)) = \frac{1}{2}(h_\theta(x^i) - y^i)^2$ - Overall Cost Function: $J_{train}(\theta) = \dfrac{1}{m}\sum_{i=1}^{m} cost(\theta, (x^i, y^i))$
- Procedure:
- Randomly shuffle dataset
- Repeat
for $i \in [1, m]$

$$\begin{aligned} \theta_j &= \theta_j - \alpha \frac{\partial}{\partial \theta_j} cost(\theta, (x^i, y^i)) \\ &= \theta_j - \alpha(h_\theta(x^i) - y^i) \cdot x^i_j \end{aligned} \tag{2.20}$$

$$\text{(for } j = 0, ..., n) \tag{2.21}$$

=> make progress with each single data (2.22)
- Convergence:
- Wanting θ to converge =¿ slowly decrease α over time (but more parameters)
(E.g $\alpha = \dfrac{\text{const\_1}}{\text{iteration num + const\_2}}$)
- Compute $cost(\theta, (x^i, y^i))$ before updating
For every k update iterations, plot average $cost(\theta, (x^i, y^i))$ over the last k examples
- Checking curves:
Increasing k result in smoother line and less noise, but the result is more delayed
Use smaller learning rate α will generally have slight benefit
Curve goes up =¿ smaller α
- *vs* Batch Gradient Descent:
- use 1 example un each update iteration =¿ make progress earlier =¿ faster - Result may not be the optimal but in its neighbourhood
1. Mini-batch Gradient Descent - Use b examples in each update iteration - *vs* Batch Gradient Descent: - start to make progress earlier =¿ faster - Result may not be the optimal but in its neighbourhood - *vs* Stochatistic Gradient Descent:t - can partially parallelize computation over b examples =¿ faster under a good vectorized implementation & appropriate b - introduce extra parameter b

2.11.2 Online Learning

1. Situation: - Has too many data (can be considered as infinite) - When data comes in as a continuous stream - Can adapt to changing user preference 2. Procedure: - Use one example only once (Similar to stochastic gradient decent in this sense

2.11.3 Map-reduce

1. In Batch Gradient Descent:
   - Update rule $\theta_j = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i$ - Parallelize the computation of $\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i$ by dividing the data set into multiple sections
   2. Ability to reduce:
   - Contain operation over the whole data set
   (Neural Network can be map-reduced)

2.12 Building Machine Learning System

- Under the example of Photo OCR (Optical Character Recognition)

2.12.1 Pipeline

1. Break ML system into modules
   2. Example:
   - Image -¿ Text detection -¿ Character segmentation -¿ Character recognition
   - Text Detection:
   - Sliding window detection:
   set different sizes of the window (mostly rectangle), for each size:
   take a image patch
   resize the patch into desired size
   run ML algorithm on the small patch
   slide the window by step_size (eventually through the image)
   - Expansion: expand the related region to create a bigger region
   - Chraracter Segmentation:
   - 1-D sliding window
   - Character Recognition

2.12.2 Getting More Data

1. **Artificial Data Synthesis** - Creating New Data: Use available resource and combine them - Example (in Character Recognition): Paste different fonts in the randomly chosen backgrounds - Amplify Data Set: Intorduce distortions to the original data set - Need to identify the appropriate distortion - Usually adding purely/random/meanless noise

   - Prerequisite: - Having a low bias/high variance hypothesis is

   1. Collect/Label Data Manually - Usually a surprise to find how little time it needs to get 10,000 data - Caculate the time it needs before decide to/not to collect the data

2.12.3 Ceilling Analysis

1. Aim:
   - Decide which modules might be the best use of time to improve
   2. Procedure:
   - Draw a table with 2 column (Component - Accuracy)
   - Component: the modules simulated to be perfect (100- Accuracy: the accuracy of the entire system on the test set (define by chosen evaluation matrix)
   - — **Perfect Component** — **Accuracy** — — :————————: — :——————————————: — — none — f — — module 1 — $f + \epsilon_1$ — — module $1,2$ — $f + \epsilon_1 + \epsilon_2$ — — ·
·
· — ·

.

· — — module $1, 2, ..., n$ — $f + \epsilon_1 + ... + \epsilon_n = 100\%$ —

- =¿ Improving module x will gain at most ϵ_x improvement in the overall performance
- Choose the module with most significant ϵ to improve

Chapter 3

Linear Regression

Chapter 4

Linear Classification

Chapter 5

Kernel Methods

Chapter 6

Graphical Models

Chapter 7

Mixture Models and EM

Chapter 8

Approximate Inference

Chapter 9

Sampling Methods

Chapter 10

Continuous Latent Variable

Chapter 11

Sequential Data

Chapter 12

Deep Learning

12.1 Interview of Fame

12.1.1 Geoffrey Hinton

Knowledge Embedding

- BP
  - psychology view: knowledge in vectors
  - semantic AI: knowledge graph
  - BP algorithm can interpret & convert between feature vector and graph representation (with some embedding)
- Boltzmann Machine
  - Leaning Algorithm on Density Net
    - same information in forward & backward propagation to learn feature embedding
  - Restricted Boltzmann Machine (RBM)
    - ways of learning in deep dense net with fast inference
    - iterative learning (adding layer after the above trained)
    - ReLU $\Leftrightarrow$ a stack of sigmoid functions (approximately) in RBM
    - ReLU units initialized to identity for efficient learning
- EM
  - EM with Approximate E Step
- vs. Symbolic AI
  - Symbolic AI: symbolic logic-like expression to do reasoning
  - yet, maybe state vector to represent knowledge

Brain Science

- Brain: Nets Implemented by Evolution
  - trying to train without BP
  - doing BP (get derivatives) with re-construction error (auto-encoder)

Memory in Nets

- Fast Weights for Short-term Memory
- Capsule Net
  - structured knowledge representation in each unit (feature with sets of property)
  - ⇒ enable nets to vote rather than filtering - thus better generalization

Unsupervised Learning

- Importance
  - better than human eventually (as supervised learning has limited maximum)
  - GAN as a breakthrough

"Slow" Feature

- Non-linear Transform to Find Linear Transform
  - find a latent representation containing linear transform to do the work
  - e.g. change viewpoints: pixels → coordinates → linear transform → back to pixels

Relations between Computers

- showing computer data to work
  - instead of programming it to work

12.1.2 Pieter Abbeel

Deep Reinforcement Learning

- Overall Challenge
  - Representation
  - Exploration Problem
  - Credit Assignment
  - Worst Case Performance
- Advantage (Deep Nets in RL)
  - network capturing the representation (state vector)
- Question in DRL
  - how to learn safely
  - how to keep learning (under small negative samples) e.g. better than human
  - can we learn the reinforcement learning program (RL in the RL)
  - long time horizon
  - use experience across tasks
- Success of DRL
  - simulated robot inventing walking... ⇒ single general algorithms to learn

12.1.3 Ian

Generative Adversarial Networks

- Generative Models
  - Resembling
    - trained to optimized the distribution behind training data
      (then sampled from that distribution to get more imaginary training data)
    - $\Rightarrow$ produce data to resemble the training data
  - Usage
    - semi-supervised learning
    - data augmentation
    - simulating scientific experiment
  - Previous Ways
    - Boltzmann Machine
    - Sparse Coding
  - Now: Generative Adversarial Networks (GANs)
  - Future
    - increase reliability of GANs (stabilizing)

12.1.4 Research

Topics

- Unsupervised Learning
- Reinforcement Learning
- AI Security
  - Anti Inducing
    - NOT to be fooled/induced to do unappropriated things
      (even if algorithm is right)
  - Built-in Security
- Fairness in AI
  - Dealing Societal Issue
  - Reflecting Preferred Bias
- **Auto Optimization (Hyperparameter Tunning)**
  - Swarm Optimization
  - Expectation Maximization
    - target variable θ = hyperparameters
    - hidden variable Z = weights of network
    - data X = dataset

    $\Rightarrow$
    - E-step: evaluate $\mathbb{E}_{Z|\theta_n,X}(\ln P(Z,X|\theta))$
      - $\ln P(Z,X|\theta)$: log likelihood of hyperparam θ (for weights & data to be observed)

  - $\cdot$ $P(Z|\theta_n, X)$: posterior of weights Z

    $\Rightarrow$ evaluate (approximate) the expectation of the log likelihood of hyperprarm θ
    (from a functional view, train with $\theta_0 - \theta_N$, evaluate model M times in training, thus with weights $Z_{00} - Z_{NM}$)
    $\Rightarrow$ a matrix with n as row entry, m as column entry, mapping to both $\ln P(Z, X|\theta), P(Z|\theta_n, X)$
    $\Rightarrow$ then marginalize (taking the expectation) over Z, to get a (sampled) function over θ
- ▪ M-step: maximize the result function from E-step
  - $\cdot$ fit a curve & maximize w.r.t hyperparams θ

Advises

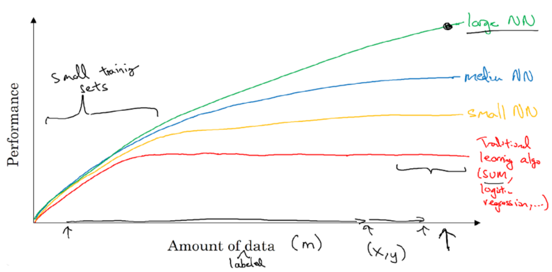
- Reading
  - ◦ read a little bit & find somewhere intuitively not right
    - ▪ good intuition: eventually work; bad intuition: not working no matter what it is doing
    - ▪ if other doubts your idea as bullshit $\Rightarrow$ a sign for real good result
  - ◦ a supervisor with similar belief
  - ◦ PhD vs. Company
    - ▪ amount of mentoring
    - ▪ faster if dedicated supervisor available
- Practice
  - ◦ open-source learning resource
  - ◦ implement the paper
  - ◦ work on a projected and open source it
    $\Rightarrow$ the stage (e.g. github) will bring people to you

12.2 Basic Neutral Network

12.2.1 Advantages

Large/Big Data

- Larger Maximum Capability
  - ◦ Curve given Amount of Data



  - ◦ Reasons
    - ▪ the scale of data (labeled)
    - ▪ the scale of neural network (computability)
    - ▪ the scale of efficiency: e.g. ReLu, faster parallel algorithm

Flexibility

- Different Structures for Different Tasks
  - Same Data & Task
    - changing settings/structures of deep learning model can make a difference (v.s. SVM, etc.)
- Ability to Choose Basis Functions
  - Functional View
    - $y(\mathbf{x}, \mathbf{w}) = f(\mathbf{w}^T \phi(\mathbf{x})),$ where ϕ is basis function , $f(\cdot)$ is net as a function
  - Learning ϕ: choose embedding $\Rightarrow$ choose basis function
  - Learning $\mathbf{w}$: choose which feature / basis functions more useful
- Solving Bias-Variance Trade-off
  - Complexity + Data/Regularization
    - easy complexity via depth, size
      $\Rightarrow$ reduce bias, without hurting variance by utilizing big data
    - easy regularization via L2 ant etc.
      $\Rightarrow$ prevent high variance without hurting bias much in a deep/big net

Power of Depth

- Deep Representation
  - Low-level $\rightarrow$ High-level
    - multiple layers to choose & combine useful information (creating new feature/basis)
      $\Rightarrow$ next layer use chosen/combined simple basis to build more complex one
    - $\Rightarrow$ an hierarchy from low-level information to high-level information
- Circuit Theory
  - Power of Combination
    - functions that can be compactly represented by a depth k architecture might require an exponential number of computational nodes using a depth $k-1$ architecture
      (from the perspective of factorization)

  Yet, start from the SHALLOW (logistic regression) before trying the deep

12.2.2 Problem

(n units in one hidden layer)

Weight-space Symmetries

- Symmetries in Activation Function
  - $\mathcal{O}(2^n)$, e.g. $\arctan(-x) = -\arctan(x) \Rightarrow$ changing signs of all input & output has the same mapping (reduce effective data)
- Positional Combination in One Layer
  - $\mathcal{O}(n!)$ exchange unit with each other (together with their input output weights) $\Rightarrow$ mapping stay the same

$\Rightarrow \mathcal{O}(n!2^n)$ overall weight-space symmetries

Non-convex Error Function

- Multiple Critical Points
  - at least $\mathcal{O}(n!2^n)$ critical points ($\nabla E(w) = 0$, where $E(w)$ is error function) due to weight-space symmetries
- Expensive in Finding Critical Point
  - expensive for even local optima with gradient decent
  - as expensive as $\mathcal{O}(n^3)$ if using Laplace approximation

Gradient Vanishing/Exploding

- Gradient Vanishing
  - Saturated Function
    - sigmoid/tanh function: gradient $\to 0$ when input $\to \pm\infty$
  - Exponential Effect
    - with depth L, each activation (e.g. tanh) output $a^l < 1$ and weight $\mathbf{w}^l < 1$
      $\Rightarrow y(\mathbf{x}, W) \approx w^L \mathbf{w}'^{L-1}\mathbf{x}$, with $\mathbf{w}' < 1$
      $\Rightarrow$ all the gradient along the way get multiplied by number < 1
      $\Rightarrow$ gradient exponentially decayed in back-prop
- Gradient Exploding
  - Exponential Effect
    - similarly, each activation (e.g. ReLU) output $a^l > 1$ and weight $\mathbf{w}^l > 1$
      $\Rightarrow y(\mathbf{x}, W) \approx w^L \mathbf{w}'^{L-1}\mathbf{x}$, with $\mathbf{w}' > 1$
      $\Rightarrow$ all the gradient along the way get multiplied by number > 1
      $\Rightarrow$ gradient exponentially augmented in back-prop
- Possible Solutions
  - Random Initialization
    - Xavier Initialization: for gradient vanishing & exploding
  - Activation
    - ReLU: for gradient vanishing

12.2.3 Learning

Forward-Backward Propagation

- Representation
  - Layers
    - input layer
    - hidden layer(s): layer with NO ground truth (for the associated weights) available
      note: input & hidden layers have associated biases as well (usually)
    - output layer
  - Neuron (Unit)
    - s_l: num of units in layer l
    - w^l: weight matrix of mapping from layer l to $l+1$, with shape of $(s_{l+1}, s_l + 1)$
    - a^l_j: activation of unit j at layer l

    - $h(\cdot)$: activation function (usually shared)
    - z^l_j: output of unit j at layer l (represent parameterized basis)
  - Intuition
    - all stacked vertically (vertical vector)
      $\Rightarrow$ horizontally for different examples; vertically for different units

- Forward Propagation (Inference)
  - Activation $a^{j+1} = w^j \cdot [z^j_0, ..., z^j_{s_j}]^T$, with $z_0 = 1$
  - Unit Output $z^{j+1} = h(a^{j+1}) = [z^1_1, ..., z^{j+1}_{s_{j+1}}]^T$
- Backward Propagation
  - Loss $\mathcal{L}(W) =$
- Practice of Back Prop
  - Caching Intermediate Result
    - naturally cached: input $a^0 = x$, weights w and bias b
    - activation input/output a/z
      (since will be used in back-prop)
  - Auto Difference
    - achievement: calculate the derivatives along the forward prop **!**

12.3 Experiment Practice

12.3.1 Tunning Hyperparameters

Hyperparameters

- Overview
  - Structures and Architectures
    - type of layers and size of layers
    - type of activation
    - depth of networks
    - so on...
  - Learning
    - learning rate
    - optimizer (learning process)
- Consequences
  - NO Consistent Prescience
  -
-

12.3.2 Designing Networks

12.4 Operations & Layers Structure

12.4.1 Operations in Network

Activations

- Sigmoid $a = \sigma(z)$
  - Pros
    - mapping to $(0, 1)$, with $\sigma(0) = 0.5$
  - Cons
    - gradient vanishing: $\sigma(z)' = \sigma(z)(1 - \sigma(z)) \Rightarrow \lim_{z \to \pm\infty} \sigma(z)' \to 0$
      (as the gradient passed through (via chain rule) $= \frac{a}{z}\frac{z}{w}$)
- Tangent $a = \tanh(z)$
  - Pros
    - empirically, almost always better than sigmoid (in hidden layers)
    - maps to $(-1, 1)$, with $\tan(0) = 0 \Rightarrow$ help centering data (0-mean)
      $\Rightarrow$ make the learning of next layer easier
  - Cons
    - still, gradient vanishing when $z \to \pm\infty$
- Rectified Linear Unit ReLu $\max(0, z)$
  - Derivation: approximated by a stack of sigmoid
    -
  - Pros
    - mitigate gradient vanishing: $\forall z > 0, a = z \Rightarrow$ learn much faster

  $\Rightarrow$ the default choice!
  - Cons
    - undefined behavior at $x = 0$ (actually, gradient becomes the sub-gradient)
    - gradient totally vanished for $x < 0$
    - $\Rightarrow$ dead units: weights learned/initialized to always output negatives
      $\Rightarrow$ activation always output 0
      $\Rightarrow$ the unit always output 0
- Leaky Relu $a = \max(\alpha z, z), \alpha \to 0^+$ (e.g. $\alpha = 0.01$)
  - Pros
    - mitigate the gradient vanishing problem for $(-\infty, +\infty)$
    - avoid dead units problem

  (yet not that popular as ReLU)
- Linear (Identity) Activation $a = z$
  - Pros
    - used in regression to output real number $\in (-\infty, +\infty)$

        - used in compression net
    - Cons
        - stacked units with linear activation $\Leftrightarrow$ single linear transformation
        - logistic regression with linear activation in hidden layer is NO more expressive than logistic regression with no hidden layer **!**

12.4.2 Operations on Network

Initialization

- Random Initialization for Weights
    - Practice
        - weights initialized to a random variable in a small range e.g. $(-0.03, 0.03)$
    - Pros
        - avoid symmetry problem:
          if identical initialization for weights $\Rightarrow$ units in same layer computing exactly same function
          $\Rightarrow$ get the same learning step propagated back
          $\Rightarrow$ then always compute exactly the same function (by induction)
        - avoid gradient vanishing: especially for gradient of sigmoid/tanh activation
    - Cons
        - NOT concern various nets: sampling in a fixed range may not work for all nets
- Xavier Initialization for Weights
    - Practice
        -
    - Pros
        - empirically much better:
- Zero Initialization for Bias
    - Reason
        - default to use 0 bias
          (can NOT used for weights as explained)

Regularization

- $L2$ Regularization
    - Understanding
        - forcing weights to be smaller
            - single node has smaller effect
            - input of activation closer to 0
              $\Rightarrow$ activation becomes more linear-alike (e.g. sigmoid, tanh)
              $\Rightarrow$ layers perform more linear-alike transformation
          $\Rightarrow$ simpler network, less able to fit extreme curly decision boundary
          (hence less able to overfit)
- $L1$ Regularization
- Dropout Regularization

- Definition
  - for each of selected units, set a drop probability
    i.e. for each forward/back-prop, nodes are "dropped" according to the probability
    ⇒ for each time, a randomly reduced net is trained
- Implementation: Inverted Dropout
  - set a keep prob k instead of drop prob, for a selected layer
  - generate random numbers for all units & turned into a boolean "keep" vector $\mathbf{k}$
  - dropped activation $\mathbf{d} = \mathbf{a} \times \mathbf{k}$ (element-wise),
    where $\mathbf{a}$ is original activation output vector from the layer
  - ⇒ activation becomes 0 for dropped units in $\mathbf{d}$
  - scaling up by dividing the keep prob: $\mathbf{d}/k$
    ⇒ so that expected output value of each activation remains the same
  - test time: no dropout ⇒ no random output
    (randomness in training, mitigated by big data)
- Understanding
  - can NOT rely on any one feature ⇒ have to spread out weights
    ⇒ results in shrinking the squared norm of weights (as $L2$)
  - used on layers with enormous features as input (e.g. computer vision)
    ⇒ reduce the chance of relying on small set of features
- Cons
  - training loss may have bigger glitch ⇒ harder to debug
    (make sure loss decreasing before introduced dropout)

- Early Stopping
  - Definition
    - stop the training at lowest validation loss (with training loss decreasing)
      ⇒ at the start point of overfitting
  - Practice
    - evaluate both train & val loss, saving models along the way
      ⇒ use the model corresponding to the start of overfitting
  - Understanding
    - at relatively early stage, weights are still relatively small
      (due to random initialization in $[0^-, 0^+]$)
  - Cons
    - couples task of optimizing loss and task of not overfitting
      ⇒ no longer one task at a time

12.4.3 Layers

Convolution Layer

- Normal Convolution
- Atrous Convolution
- Deconvolution

Pooling Layer

- Normal Pooling
- Unpooling
- Spatial Pyramid Pooling (SPP)
- Region of Interest Pooling (RoI Pooling)
  - Input
    - feature maps from CNN
    - RoIs i.e. proposal region (from selective search etc.) projected on feature map
  - Operation
    - divide each RoI with grid of desired size (proportional to the RoI size)
    - max pooling from each cell

    ⇒ single-size SPP for each RoI
  - Output
    - a fixed size feature maps for each RoI

12.5 Architectures

12.5.1 Encoder-Decoder Architecture

Description

- Encoder
  - Functionality
    - downsample/encode input into rich feature maps/vectors
  - Implementation
    - visual input: CNN backbone
    - natural expression input: RNN backbone
- Decoder
  - Functionality
    - upsample/decode rich feature maps back to the original size
    - actually, impose requirement onto the encoder
  - Implementation
    - visual output: CNN backbone
    - natural expression output: RNN backbone
- Connection
  - Functionality
    - combine high level information with low level information
    - image → image: outline refinement ...
    - language → language: sentence style capturing
  - Implementation
    - concatenation

Extension

- Multiple Encoder
  - Functionality
    - project different information into the same space
    - combine those information via some shared layers at the end
- Multiple Decoder
  - Functionality
    - impose multiple requirements to the encoder (via auxiliary loss)

12.6 Advanced Topic

12.6.1 Machine Reading Comprehension

Problem Formulation

-

RNN with Attention

Convolution with Self-attention - QAnet

12.6.2 Image Caption

Problem Formulation

- Input
  - Image
    - visual input as the target of description
- Goal
  - Natural Expression
    - description of the image in natural language, e.g. English

Baseline Approach & Previous Work

- Neural Image Caption
  - Visual Information
    - encoded by CNN backbone into a 1-D vector
  - Word Information
    - a set of word selected beforehand
    - word embedding performed
  - Language Generation
    - generated by an LSTM decoder
    - combining info: visual encoding as initial state of LSTM
    - process: LSTM gives each word a to-be-selected probability at each time step
  - Inference
    - sampling: sample each word according to the distribution given by LSTM
    - beam search: iteratively consider extending k best sentence of length t to $t+1$ $\Rightarrow$ select k best sentence of length $t+1$ from all resulted sentences

    (beam search selected in the paper)

12.6.3 Referring Segmentation

Problem Formulation

- Input
  - Image
    - visual input for segmentation
  - Natural Language Expression
    - expression to denote the interested object(s)/stuff(s)
- Goal
  - Segmentation Mask of Referred Object(s)
    - currently (till early 2019), mostly binary segmentation
- Related Area
  - NLP + CV
    - referring localization
    - image caption

Baseline Approach & Previous Work

- Segmentation from Natural Language Expressions
  - Spatial Info
    - FCN-32s to encode the image into 2-D feature maps (the last conv layer)
  - Language Info
    - LSTM to encode the sentence into 1-D vector (the last hidden state)
  - Combining Info and Output
    - per-pixel info: concat [coordinates of current pixel (coord info), language info]
    - tile the per-pixel info into a feature map, then concat to the spatial info (per-pixel info concatenated at every pixel of spatial info)
    - followed by a series of conv and finally a deconv for upsampling
  - Training
    - per-pixel cross-entropy loss
  - Pros
    - special spatial info: coord of each pixel
    - standard info combination: concatenation
  - Cons
    - no powerful spatial info encoder: FCN-32s instead of Resnet/Unet...
    - weak upsampler, compared to encoder-decoder architecture
    - language info comes late: after downsampling
    - weak language info: only integrated once
- Recurrent Multimodal Interaction for Referring Image Segmentation
  - Spatial Info
    - DeepLab-101 as encoder (Resnet as backbone, with atrous conv)
    - then tiled (concat at every pixel) by coord info (coordinate of current pixel)

- Language Info
  - word embedding w_t for $t = 1, ..., T$
  - LSTM scanning the sentence, with hidden state h_t at time t
  - language info l_t = concat $[h_t,\, w_t]$
- Combining Info
  - l_t tiled to spatial info, at each time step
    $\Rightarrow$ creating combined feature maps F_t (of shape $[\text{height}, \text{wide}, \text{channel}]$)
  - combined feature maps $F_1, ..., F_T$ fed to an convolutional LSTM,
    where the ConvLSTM shares weight over both space and time
    $\Rightarrow$ feature vector of $F_t[i, j]$ is the input of the ConvLSTM at time t
    $\Rightarrow$ conv in ConvLSTM implemented as 1×1 conv
  - a series of conv following the last hidden state of the ConvLSTM
- Output
  - bilinear interpolated to original input size
  - optionally post-processed by dense CRF, using pydensecrf (hence inference only)
- Pros
  - more powerful spatial info extractor: DeepLab-101
  - better language info: integrated at every time step, maintained by an ConvLSTM
- Cons
  - weak architecture for spatial info: still no upsampling (blur segmentation)
  - no spatial relation considered in ConvLSTM (?)
  - weak language representation
    (better with pos tag, word2vec, word dict, biLSTM, and maybe even attention)
  - language info still comes late: still after downsampling

Current State-of-The-Art (early 2019)

- Key-Word-Aware Network for Referring Expression Image Segmentation
  - Spatial Info
    - DeepLab-101 as encoder for comparability
    - then tiled by coord info (coordinate of each pixel)
  - Language Info
    - LSTM scanning sentence, each hidden state as word info
  - Combining Info
    - attention mask from combined info (spatial info with language info tiled)
      (at each time step)
    - attention weighting over spatial info at each time step
      $\Rightarrow$ an 1-D global encoding for each time step (via weighted mean over space)
      $\Rightarrow$ filling feature maps: global encoding if attention here $>$ threshold; else $\mathbf{0}$
      $\Rightarrow$ summing filled feature maps over time for the global spatial maps c
    - attention weighting over tiled language info at each time step, correspondingly
      $\Rightarrow$ tiled language info maps summed over time for the global language maps q
    - concat [spatial info, c, q], followed by 1×1 conv
  - Output
    - upsampling performed

- Pros
  - attention introduced: from combined info
  - better combination: attention masked interact with both spatial & language info
- Cons
  - blur segmentation: no encoder-decoder architecture
  - attention mask obtained sequentially: only last mask has complete language info
  - language info comes late: after downsampling

• Referring Image Segmentation via Recurrent Refinement Networks

- Spatial Info
  - DeepLabl ResNet-101 as encoder
  - last feature maps tiled (concat at each pixel) with coord info
- Language Info
  - LSTM scanning sentence, generating word info at each time step
  - last hidden layer as language info
- Combining Info
  - combined info = spatial info tiled with language info
  - selecting set of feature maps from downsampling stages
  - all selected feature maps resized and fed to 1×1 conv
    $\Rightarrow$ to match the dimensions of combined info
  - convolutional LSTM applied to refine the combined info
    (with matched selected feature maps as input at each time step)
- Output
  - a conv after final hidden state of ConvLSTM for segmentation
  - upsampled to original image size
- Pros
  - ConvLSTM integrating info at dowsampling stage $\Rightarrow$ segmentation refined
- Cons
  - no upsampling: blur segmentation, mitigated by ConvLSTM though
    (yet no language info introduced in refinement)
  - CNN fixed during training: relying on ConvLSTM
  - single info combination: only by tiling
    (though, currently performing the best in all dataset)

Research Direction

• Integrating Encoder-Decoder Architecture

- Upsampling
  - similar to Unet, concat low-level spatial info
  - introduce language info as well
    (e.g. early combination, explicit introducing, ...)

• Early Info Combination

- Tiling at First Conv
  - downsampling more responsible for language info processing
    $\Rightarrow$ hopefully get more fine-tuning alone with conv filters

    - can be used with pre-trained net:
      $ReLU(conv_1 * X_1 + conv_2 * X_2) = ReLU([conv_1, conv_2] * [X_1, X_2])$
  - Multiple Entries
    - combining info at different stages of downsampling / upsampling

- Attention
  - Attention from Combined Info
    - as key-word-aware net
  - Attention on Language Info
    - 1-D spatial pyramid pooling / attention mask on the sentence encoding

- Language Info Throughout Network
  - Encoder-Decoder for Language Info
    - network asked to recover language info after processing combined info
      (potentially via a separate branch only at training time)
      $\Rightarrow$ auxiliary loss
  - Language as Conv Filter
    - Language Info, through a subnet, becoming a set of conv filters
      $\Rightarrow$ then imposed in downsampling, tunnel, upsampling or bridge stage(s)

- Data Augmentation
  - Translation Module
    - using the same image
    - expression translated to a middle language and then back to English
      $\Rightarrow$ language info trained more finely