

Machine Learning

Liyao Tang

February 14, 2019

Contents

1	Math	1
1.1	Convolution	1
1.2	Linear Algebra	1
1.2.1	Essence	1
1.2.2	Interchanging Coordinates	7
1.3	Calculus	10
1.3.1	Integral	10
1.4	Probability Theory	11
1.4.1	Introduction	11
1.4.2	Expectations and Covariances	13
1.4.3	Transformations of Random Variables	14
1.4.4	Gaussian Distribution	15
1.4.5	Bayesian Interpretation of Probability	18
2	Introduction	21
2.1	General Concern	21
2.1.1	Types of Learning	21
2.2	Model Selection	22
2.2.1	Project Structuring	22
2.2.2	Over-/Under-fitting	22
2.3	Decision Theory	23
2.3.1	23
2.4	Information Theory	23
2.5	Supervised Learning	23
2.6	Linear Regression	23
2.7	Bayesian Regression	25
2.8	Logistic Regression (Classification)	27
2.9	Diagnose Machine Learning	31
2.9.1	Evaluating Hypothesis	31
2.9.2	Bias / Variance	31
2.9.3	Error Analysis	32
2.9.4	Skewed classes	32
2.10	Latent Variable Analysis	32
2.10.1	Principal Component Analysis (PCA)	32
2.10.2	Independent Component Analysis (ICA)	35
2.10.3	t-SNE	36
2.10.4	Anomaly Detection	36
2.10.5	Recommender System	36
2.11	Large Scale Machine Learning	38
2.11.1	Gradient Descent with Large Dataset	38
2.11.2	Online Learning	39
2.11.3	Map-reduce	39

2.12 Building Machine Learning System	39
2.12.1 Pipeline	39
2.12.2 Getting More Data	39
2.12.3 Ceilling Analysis	39
3 Linear Regression	41
4 Linear Classification	42
5 Kernel Methods	43
6 Graphical Models	44
7 Mixture Models and EM	45
8 Approximate Inference	46
9 Sampling Methods	47
10 Continuous Latent Variable	48
11 Sequential Data	49
12 Deep Learning	50
12.1 Interview of Fame	50
12.1.1 Geoffrey Hinton	50
12.1.2 Pieter Abbeel	51
12.1.3 Research Advices	52
12.2 Basic Neutral Network	52
12.2.1 Goal and Advantages	52
12.2.2 Problem	53
12.2.3 Learning	54
12.3 Advanced Topic	54
12.3.1 Machine Reading Comprehension	54
12.3.2 Referring Segmentation	54

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

$$\begin{aligned} H(z) &= P(Z < z) = P(X + Y < z) \\ &= \int_x P(X = x)P(X + Y < z|X = x)dx \\ &= \int_x f(x)P(Y < z - x)dx \\ &= \int_x f(x)G_Y(z - x)dx \\ \Rightarrow h(x) &= \frac{d}{dz}H(z) = \frac{d}{dz} \int_x f(x)G_Y(z - x)dx \\ &= \int_x f(x) \frac{dG_Y(z - x)}{dz} dx \\ &= \int_x f(x)g(z - x)dx \\ &= f * g(z) \end{aligned}$$

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
 $2\mathbf{x} = 2x_1e_1 + \dots + 2x_ne_n$,
 where e_1, \dots, 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: $\mathbf{x} = a_1\mathbf{x}_1 + \dots + a_n\mathbf{x}_n$
 - the vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ only altered linearly (as only being scaled)
 \Rightarrow \mathbf{x} direction & size are linear combination of that in $\mathbf{x}_1, \dots, \mathbf{x}_n$
 - Span of $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$
 - the n-D space S constructed by linear combination of $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$
 - x_0 linearly dependent on $\{x_1, \dots, x_n\}$
 - x_0 can be constructed by linear combination of $\{x_1, \dots, x_n\}$
 (already in the span space S)
 - \Leftrightarrow function $a_0x_0 + \dots + a_nx_n = 0$ has other solution than $a_0 = \dots a_n = 0$
 - x_0 linearly INdependent with $\{x_1, \dots, x_n\}$
 - x_0 can NOT be constructed by linear combination of $\{x_1, \dots, x_n\}$
 (not in the span space S , will increase the dimension of S if adopted)
 - \Leftrightarrow function $a_0x_0 + \dots + a_nx_n = 0$ has and only has solution $a_0 = \dots a_n = 0$
 - 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

Matrix

- 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
 - 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 scalar
 - Features given $\mathbf{x} = x_1e_1 + \dots + x_ne_n$
 - same scalar for coordinates

$$\begin{aligned} \Rightarrow L(\mathbf{x}) &= L(x_1e_1 + \dots + x_ne_n) \\ &= L(x_1e_1) + \dots + L(x_ne_n) \\ &= x_1L(e_1) + \dots + x_nL(e_n) \end{aligned}$$

$$\Rightarrow \text{transformed vector } \mathbf{x}' = L(\mathbf{x}) \text{ has the same coord under the transformed basis}$$
- Matrix for Linear Transformation $L : S \rightarrow S'$
 - Representing Space Transformation
 - package the transformed basis under the original basis using matrix M
i.e. represent the $e'_1, \dots, e'_n = L(e_1), \dots, L(e_n)$ under the original basis e_1, \dots, e_n
 $\Rightarrow M = [e'_1, \dots, e'_n]$, 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 $L : S \rightarrow S'$ using e_1, \dots, 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
 - $\forall i \in \{1, \dots, n\}, x'_i = i^{\text{th}}$ component of $\mathbf{x}' = L(\mathbf{x})$, then $x'_i = (e'_{1i} + \dots + e'_{ni})x_i$
(as proved above)
 - \Rightarrow output vector $\mathbf{x}' = L(\mathbf{x}) = M\mathbf{x}$ under the original basis e_1, \dots, 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 $E = [e_1, \dots, e_n]$, a transformed basis $L_1(E) = L_1(e_1), \dots, L_1(e_n)$
 $(e_1, \dots, e_n \text{ as column vectors})$
 $\Rightarrow L_2(L_1(e_k))$ 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 $L_1(E) = M_1 \Rightarrow L_2(L_1(E)) = M_2 \cdot M_1$
 $(\text{due to the derivation of linear transformation as matrix})$
- hence, $M_2 \cdot M_1$ denotes
 1. a further L_2 transformation on a transformed space $L_1(E)$
 $(\text{all result represented under the original basis } E)$
 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 transformations into single linear transformation
 - \Rightarrow linear transformation on vectors/space (generalized from single vector)

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

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 $E = [e_1, e_2]$)

• Understanding Properties

- $(AB)C = A(BC)$
 - both meaning apply transformation C then B then A ...

Dot Product

- Projecting to 1-D
 - Unit Orthogonal Basis
 - $\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + \dots + x_n y_n = \mathbf{x}^T \cdot \mathbf{y} = \mathbf{y}^T \cdot \mathbf{x}$
 $(\mathbf{x}, \mathbf{y} \text{ assumed to be column vector / matrix with single column})$
 - General Basis $E = [e_1, \dots, e_n]$
 - $\Rightarrow \mathbf{x} = x_1 e_1 + \dots + x_n e_n = E\mathbf{x}, \mathbf{y} = y_1 e_1 + \dots + y_n e_n = E\mathbf{y}$
 $\Rightarrow \mathbf{x} \cdot \mathbf{y} = (E\mathbf{x}) \cdot (E\mathbf{y}) = \mathbf{x}^T E^T E \mathbf{y}$
 - 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
 $(\text{hence the vector equivalent to the matrix with 1 row defining the projection})$
 - \Rightarrow performing transformation on a vector \Leftrightarrow taking product with the dual vector

Determinant

- Measuring Linear Transformation on Volume
 - Unit Volume
 - unit volume $v = \|e_1 \wedge \dots \wedge e_n\|$ with basis e_1, \dots, e_n
(for orthogonal basis, $v = \|e_1\| \times \dots \times \|e_n\|$)
 - after transformation: $L(v) = \|L(e_1) \wedge \dots \wedge L(e_n)\| = \|Me_1 \wedge \dots \wedge Me_n\|$
where \wedge is the exterior product
 - Measuring Change of Unit Volume
 - $\Rightarrow \det(M) = \frac{L(v)}{v} = \|M_0 \times \dots \times M_n\|$,
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 axes 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
 - $\det(M_1 M_2) = \det(M_1) \det(M_2)$:
 - 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
 - $R(M_{m \times n}) \leq \min\{m, n\}$
 - $m < n$: 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, \dots, e_N : a group of orthonormal basis of E_N
 - $\mathbf{x} = (x_1, x_2, \dots, x_N)$: vector in E_n
 - \mathbf{x}_{i-N} : projection of \mathbf{x} onto subspace spanned by e_i, \dots, 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, \dots, e_N :

$$\begin{aligned}
& \blacksquare \cos \phi_1 = \frac{\mathbf{x}e_1}{\|\mathbf{x}\|\|e_1\|} = \frac{x_1}{r} \\
& \Rightarrow x_1 = r \cos \phi_1 \\
& \Rightarrow \mathbf{x} = r \cos \phi_1 e_1 + \sum_{n=2}^N x_n e_n
\end{aligned}$$

◦ Space e_2, \dots, e_N :

$$\begin{aligned}
& \blacksquare \text{ from above: } \mathbf{x}^2 = r^2 \cos^2 \phi_1 + \sum_{n=2}^N x_n^2 = r^2 \\
& \Rightarrow \sum_{n=2}^N x_n^2 = r^2 \sin^2 \phi_1 \\
& \blacksquare \mathbf{x}_{2-N} = \sum_{n=2}^N x_n e_n \\
& \Rightarrow \begin{cases} \|\mathbf{x}_{2-N}\|^2 = \sum_{n=2}^N x_n^2 = r^2 \sin^2 \phi_1 = r_2^2 \\ \cos \phi_2 = \frac{\mathbf{x}_{2-N} \cdot e_2}{\|\mathbf{x}_{2-N}\|\|e_2\|} = \frac{x_2}{r_2} \end{cases} \\
& \Rightarrow \begin{cases} r_2 = r \sin \phi_1 \\ x_2 = r_2 \cos \phi_2 = r \sin \phi_1 \cos \phi_2 \end{cases} \quad (\text{ as } \phi_1 \in [0, \pi]) \\
& \Rightarrow \mathbf{x}_{2-N} = r \sin \phi_1 \cos \phi_2 e_2 + \sum_{n=3}^N x_n e_n
\end{aligned}$$

◦ Space e_3, \dots, e_N :

$$\begin{aligned}
& \blacksquare \text{ from above: } \mathbf{x}_{2-N}^2 = r^2 \sin^2 \phi_1 \cos^2 \phi_2 + \sum_{n=3}^N x_n^2 = r^2 \sin^2 \phi_1 \\
& \Rightarrow \sum_{n=3}^N x_n^2 = r^2 \sin^2 \phi_1 \sin^2 \phi_2 \\
& \blacksquare \mathbf{x}_{3-N} = \sum_{n=3}^N x_n e_n \\
& \Rightarrow \begin{cases} \|\mathbf{x}_{3-N}\|^2 = \sum_{n=3}^N x_n^2 = r^2 \sin^2 \phi_1 \sin^2 \phi_2 = r_3^2 \\ \cos \phi_3 = \frac{\mathbf{x}_{3-N} \cdot e_3}{\|\mathbf{x}_{3-N}\|\|e_3\|} = \frac{x_3}{r_3} \end{cases} \\
& \Rightarrow \begin{cases} r_3 = r \sin \phi_1 \sin \phi_2 \quad (\text{ as } \phi_1, \phi_2 \in [0, \pi]) \\ x_3 = r_3 \cos \phi_3 \end{cases} \\
& \Rightarrow \mathbf{x}_{3-N} = r \sin \phi_1 \sin \phi_2 e_3 + \sum_{n=4}^N x_n e_n
\end{aligned}$$

• Proof for x_i

◦ Procedure

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

• Induction

◦ Goal

$$\blacksquare \forall i \geq 2, r_i = r \prod_{j=1}^{i-1} \sin \phi_j$$

◦ Base Case ($i = 2$)

$$\blacksquare \text{ as in observation, } r_2 = r \sin \phi_1 = r \prod_{j=1}^{2-1} \sin \phi_j$$

◦ Step Case

$$\blacksquare \text{ assumption } r_i = r \prod_{j=1}^{i-1} \sin \phi_j$$

$$\blacksquare \text{ procedure: } \mathbf{x}_{i-N} = \sum_{n=i}^N x_n e_n = r_i \cos \phi_i + \sum_{n=i+1}^N x_n e_n$$

$$\Rightarrow \|x_{i-N}\|^2 = r_i^2 \cos^2 \phi_i + \sum_{n=i+1}^N x_n^2 = r_i^2$$

$$\Rightarrow \|x_{i+1-N}\|^2 = \sum_{n=i+1}^N x_n^2 = r_i^2 \sin^2 \phi = r_{i+1}^2$$

$$\Rightarrow r_{i+1} = r_i \sin \phi_i = r \prod_{j=1}^i \sin \phi_j$$

• Derivation

◦ x_i from Combined Proofs

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

◦ Last 2 Dimensions

$$\blacksquare \mathbf{x}_{(N-1)-N} = x_{N-1} \cdot e_{N-1} + x_N \cdot e_N, \text{ with } r_{N-1} = r \prod_{j=1}^{N-2} \sin \phi_j$$

$$\Rightarrow \|\mathbf{x}_{(N-1)-N}\| = f(\phi_{N-1}, \phi_N) = r_{N-1}$$

$$\Rightarrow \phi_{N-1}, \phi_N \text{ not independent!}$$

$$(\text{actually, if } e_N = e_{N-1} + \frac{\pi}{2}, \text{ then } \phi_N = \phi_{N-1} - \frac{\pi}{2})$$

$$\Rightarrow \text{define } \theta \in [0, 2\pi) \text{ instead of } \phi_{N-1}, \phi_N \in [0, \pi]$$

$$\Rightarrow x_{N-1} = r_{N-1} \sin \theta, x_N = r_{N-1} \cos \theta \text{ (interchangeable)}$$

◦ Final Spherical Coordinates

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

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'
- $T : \begin{cases} x = x(u, v), \\ y = y(u, v) \end{cases} : \text{transformation from } D \text{ to } D'$

- Assumption

- $f(x, y)$ continuous in D
- transformation T 's partial 1st order derivatives continuous on D'
- transformation T 's Jacobian $J(u, v) = \frac{\partial(x, y)}{\partial(u, v)} \neq 0$
- transformation $T : D \rightarrow D'$ is 1-1 mapping

- Derivation

- take infinitely small square in D' :

$$\begin{array}{ll} M_4'(u, v + \delta v), & M_3'(u + \delta u, v + \delta v), \\ M_1'(u, v), & M_2'(u + \delta u, v) \end{array}$$

\Rightarrow after transformation to D :

$$\begin{array}{ll} M_4(x(u, v + \delta v), y(u, v + \delta v)), & M_3(x(u + \delta u, v + \delta v), y(u + \delta u, v + \delta v)), \\ M_1(x(u, v), y(u, v)), & M_2(x(u + \delta u, v), y(u + \delta u, v)) \end{array}$$

$$\Rightarrow x_2 - x_1 = x(u + \delta u, v) - x(u, v) = \frac{\partial x}{\partial u} \Big|_{(u, v)} \delta u$$

$$x_4 - x_1 = x(u, v + \delta v) - x(u, v) = \frac{\partial x}{\partial v} \Big|_{(u, v)} \delta v$$

$$y_2 - y_1 = y(u + \delta u, v) - y(u, v) = \frac{\partial y}{\partial u} \Big|_{(u, v)} \delta u$$

$$y_4 - y_1 = y(u, v + \delta v) - y(u, v) = \frac{\partial y}{\partial v} \Big|_{(u, v)} \delta v$$

as $\delta u, \delta v \rightarrow 0$, curvilinear boundary quadrilateral $M_1 M_2 M_3 M_4 \rightarrow$ parallelogram

$$\begin{aligned} \Rightarrow S_{M_1 M_2 M_3 M_4} &= |\overrightarrow{M_1 M_2} \times \overrightarrow{M_1 M_4}| = \left| \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_4 - x_1 & y_4 - y_1 \end{vmatrix} \right| \\ &= \left| \begin{vmatrix} \frac{\partial x}{\partial u} \delta u & \frac{\partial y}{\partial u} \delta u \\ \frac{\partial x}{\partial v} \delta v & \frac{\partial y}{\partial v} \delta v \end{vmatrix} \right| = \left| \begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} \end{vmatrix} \right| \delta u \delta v \\ &= |J(u, v)| \delta u \delta v \end{aligned}$$

- \Rightarrow infinitely small area $\delta\sigma = dxdy = |J(u, v)| \delta u \delta v$

$$\Rightarrow \int \int_D f(x, y) dxdy = \int \int_{D'} f(x(u, v), y(u, v)) |J(u, v)| du dv$$

- Integral in Cartesian \rightarrow Polar

- Result

- $dx dy = r dr d\theta$

- Derivation

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

Gaussian Integral

- Gaussian Function

- $f(x) = e^{-a(x+b)^2}$

- special form: $f(x) = e^{-(x)^2}$
- alternative form: $f(x) = e^{ax^2+bx+c}$
- no indefinite integral $\int_a^b e^{-x^2}$
- only definite integral $\int_{-\infty}^{+\infty} e^{-x^2}$

- Direct Integral

- $$\begin{aligned} \left(\int_{-\infty}^{+\infty} e^{-a(x+b)^2} dx \right)^2 &= \int_{-\infty}^{+\infty} e^{-a(x+b)^2} dx \int_{-\infty}^{+\infty} e^{-a(y+b)^2} dy \\ &= \int \int_{-\infty}^{+\infty} e^{-a[(x+b)^2+(y+b)^2]} d(x+b) d(y+b) = \int \int_{-\infty}^{+\infty} e^{-a(x^2+y^2)} dx dy \\ &= \int_0^{2\pi} \int_0^{+\infty} e^{-ar^2} r dr d\theta \\ &= \frac{\pi}{a} \end{aligned}$$

$$\Rightarrow \int_{-\infty}^{+\infty} e^{-a(x+b)^2} dx = \sqrt{\frac{\pi}{a}}, \text{ alternatively } \int_{-\infty}^{+\infty} e^{ax^2+bx+c} dx = \sqrt{\frac{\pi}{-a}} \cdot e^{\frac{b^2}{4a}+c}$$

- Even Moment of Gaussian Function

- $$\begin{aligned} \int_{-\infty}^{+\infty} x^{2n} e^{-ax^2} dx &= (-1)^n \int_{-\infty}^{+\infty} \frac{\partial^n}{\partial a^n} e^{-ax^2} dx \\ &= (-1)^n \frac{\partial^n}{\partial a^n} \int_{-\infty}^{+\infty} e^{-ax^2} dx && \text{by parameter differentiation} \\ &= (-1)^n \sqrt{\pi} \frac{\partial^n}{\partial a^n} a^{-\frac{1}{2}} \\ &= \sqrt{\frac{\pi}{a}} \frac{(2n-1)!!}{(2a)^n}, && \text{where !! is double factorial} \end{aligned}$$

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 $\lim_{\delta x \rightarrow 0} P(X \in (x, x + \delta x)) = \lim_{\delta x \rightarrow 0} p(x)\delta x \Rightarrow P(X \in (a, b)) = \int_a^b p(x)dx$
 - $\Rightarrow p(x) \geq 0$ and $\int_{-\infty}^{+\infty} p(x) = 1$
 - $\Rightarrow P(z) = \int_{-\infty}^z p(x)dx$
- Basic Rules
 - Sum Rule
 - $P(X) = \sum_Y P(X, Y)$, 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

- $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(X|Y)P(Y)}{\sum_Y P(X|Y)P(Y)}$, where Y are discrete
- $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(X|Y)P(Y)}{\int_Y P(X|Y)P(Y)}$, where Y are continuous
- Interpretation of Bayes
 - Normalization
 - the \sum, \int can be interpreted as a **normalization constant**
 $\Rightarrow \text{posterior} \propto \text{likelihood} \times \text{prior}$
 - 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 $p(x)$: $\mathbb{E}_p[f] \simeq \frac{1}{N} \sum_{n=1}^N f(x_n)$
 (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 $f(x)$:
 - $\text{var}[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2]$

$$= \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$$
 - Covariance

Covariance

- Definition
 - between Variables x, y
 - $\text{cov}[x, y] = \mathbb{E}_{x,y}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])]$

$$= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y]$$
 - between Vectors \mathbf{x}, \mathbf{y} (column vectors)
 - $\text{cov}[\mathbf{x}, \mathbf{y}] = \mathbb{E}_{\mathbf{x}, \mathbf{y}}[(\mathbf{x} - \mathbb{E}[\mathbf{x}]) \cdot (\mathbf{y}^T - \mathbb{E}[\mathbf{y}^T])]$

$$= \mathbb{E}_{\mathbf{x}, \mathbf{y}}[\mathbf{x}\mathbf{y}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^T]$$

(pairwise covariance between components of \mathbf{x}, \mathbf{y})
 - within Vector \mathbf{x}
 - $\text{cov}[\mathbf{x}] \equiv \text{cov}[\mathbf{x}, \mathbf{x}]$

(pairwise covariance between its components)
- Independence Variable
 - Independent x, y
 - $\text{cov}[x, y] = \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y] = 0$

1.4.3 Transformations of Random Variables**Inverse Image**

- Definition
 - Notation
 - function $g : \mathbb{R} \rightarrow \mathbb{R}$
 - set A in \mathbb{R}
 - Inverse Image on Set A
 - $g^{-1}(A) = \{x \in \mathbb{R} | g(x) \in A\}$

$$\Leftrightarrow x \in g^{-1}(A) \text{ if and only if } g(x) \in A$$

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 $\{-n, \dots, n\}$ 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

- $\mathbb{E}[x] = \int_{-\infty}^{+\infty} \mathcal{N}(x|\mu, \sigma^2) x dx = \mu$

- $$\Rightarrow \mathbb{E}[x^2] = \int_{-\infty}^{+\infty} \mathcal{N}(x|\mu, \sigma^2) x^2 dx$$

$$= \frac{1}{(2\pi\sigma^2)^{1/2}} \int_{-\infty}^{+\infty} x^2 \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} dx$$

$$= \pi^{-\frac{1}{2}} \int_{-\infty}^{+\infty} (\sqrt{2\sigma^2}x + \mu)^2 \exp(-x^2) dx, \text{ substituting } a = \frac{x-\mu}{\sqrt{2\sigma^2}}$$

$$= \pi^{-\frac{1}{2}} (2\sigma^2 \int_{\mathbb{R}} x^2 e^{-x^2} dx + 2\mu\sqrt{2\sigma^2} \int_{\mathbb{R}} x e^{-x^2} dx + \mu^2 \int_{\mathbb{R}} e^{-x^2} dx)$$

$$= \pi^{-\frac{1}{2}} (2\sigma^2 \int_{\mathbb{R}} x^2 e^{-x^2} dx + 2\mu\sqrt{2\sigma^2} \cdot 0 + \mu^2 \sqrt{\pi})$$

$$= 2\sigma^2 \pi^{-\frac{1}{2}} \int_{\mathbb{R}} x^2 e^{-x^2} dx + \mu^2$$

$$= \sigma^2 + \mu^2, \text{ by 2nd moment of Gaussian or } (xe^{-x^2})' = e^{-x^2} - 2x^2 e^{-x^2}$$

- Variance

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

- Multivariate (d -dimensional) Gaussian

- Variable

- mean: $\mu \in \mathbb{R}^d$
 - covariance matrix: $\Sigma_{d \times d}$

- Probability Dense Function (PDF)

- $\mathcal{N}_d(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right\},$
noted as $X \sim \mathcal{N}_d(x|\mu, \Sigma)$

Multivariate Gaussian

- Dimensionality

- Volume of High Dimensional Sphere

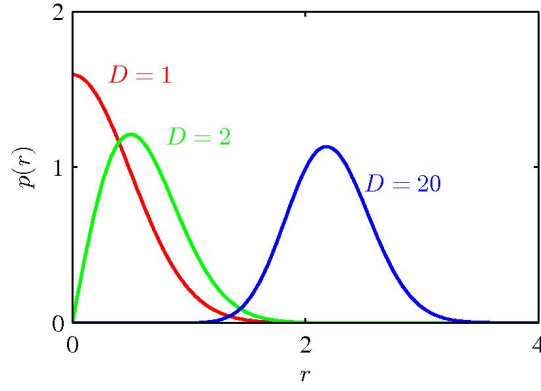
- for $n = 2k, k \in \mathbb{N}^+, V_{2k}(R) = \frac{\pi^k}{k!} R^{2k}$
 - for $n = 2k + 1, k \in \mathbb{N}, V_{2k+1}(R) = \frac{2^{k+1} \pi^k}{(2k+1)!!} R^{2k+1}$
 - $\Rightarrow \lim_{D \rightarrow +\infty} \frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = \lim_{D \rightarrow +\infty} 1 - (1-\epsilon)^D = 1 \Rightarrow$ 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 $\int G_1 G_2 dx$

- $G_1 \sim \mathcal{N}_d(x|a, A), G_2 \sim \mathcal{N}_d(x|b, B)$

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

○ \Rightarrow Convolution of Gaussians $G_1 * G_2$

$$\blacksquare G_1 \sim \mathcal{N}_d(a, A), G_2 \sim \mathcal{N}_d(b, B)$$

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

1.4.5 Bayesian Interpretation of Probability

Contrasting Frequentist Estimator

- Posterior $p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$
 - 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 $\Rightarrow \prod \rightarrow \sum$
- numerical stability \Rightarrow avoid \prod (small probabilities)
(may otherwise underflow the numerical precision)
- Maximum Likelihood Estimation for Gaussian
 - Notation
 - $X = \{x_1, \dots, x_N\}$: observed N data points
 - Assumption
 - data points are i.i.d. (identically and independently distributed)
 - underlying distribution is Gaussian $\mathcal{N}(\mu, \sigma)$
 - Likelihood
 - $p(X|\mu, \sigma^2) = \prod_{n=1}^N p(x_n|\mu, \sigma^2)$
 - $\Rightarrow \ln p(X|\mu, \sigma) = -\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi)$
 - 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 $\frac{\partial}{\partial \sigma^2} \ln p(X|\mu, \sigma^2) = 0 \Rightarrow \sigma_{\text{ML}}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu_{\text{ML}})^2$
 - Analysis
 - $\mathbb{E}[\mu_{\text{ML}}] = \mathbb{E}\left[\frac{1}{N} \sum_{n=1}^N x_n\right] = \frac{1}{N} \sum_{n=1}^N \mathbb{E}[x_n] = \mu$
(as x_1, \dots, x_N i.i.d, drawn from $\mathcal{N}(\mu, \sigma^2)$, thus $\sim \mathcal{N}(\mu, \sigma^2)$)
 - \Rightarrow unbiased estimation of mean
 - $\mathbb{E}[\sigma_{\text{ML}}^2] = \mathbb{E}\left[\frac{1}{N} \sum_{n=1}^N (x_n - \mu_{\text{ML}})^2\right]$

$$= \mathbb{E}\left[\frac{1}{N} \sum_{n=1}^N (x_n^2 - 2\mu_{\text{ML}}x_n + \mu_{\text{ML}}^2)\right]$$

$$= \frac{1}{N} \sum_{n=1}^N \mathbb{E}[x_n^2] - \mathbb{E}\left[\frac{1}{N} \sum_{n=1}^N 2x_n\mu_{\text{ML}}\right] + \mathbb{E}[\mu_{\text{ML}}^2]$$

$$= \frac{1}{N} \sum_{n=1}^N \mathbb{E}[x_n^2] - 2\mathbb{E}[\mu_{\text{ML}}^2] + \mathbb{E}[\mu_{\text{ML}}^2]$$

$$= \frac{1}{N} \sum_{n=1}^N \mathbb{E}[x_n^2] - \frac{1}{N^2} \sum_{i,j=1}^N \mathbb{E}[x_i x_j]$$

$$= \frac{1}{N} \sum_{n=1}^N (\sigma^2 + \mu^2) - \frac{1}{N^2} [N(N-1)\mu^2 + N(\sigma^2 + \mu^2)]$$

(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 !

$$\Rightarrow \text{unbiased variance } \hat{\sigma}^2 = \frac{N}{N-1} \sigma_{\text{ML}}^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \mu_{\text{ML}})^2$$

interpretation: $N - 1$ degree of freedom,

(as calculating σ^2 needs μ , which help pin down x_N given x_1, \dots, 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 Model Selection

2.2.1 Project Structuring

Data Arrangement

- Train-Val-Test
 - Training Set
 - data used to find the model parameter estimation
⇒ over-fit by complex model
 - Validation Set
 - data to indicate generalization ability of a range of models
⇒ for model comparison & selection
 - may still **over-fit** if model design iterated for too many times!
 - Test Set
 - to indicate the generalization ability of finally selected model
- Constraints on Val
 - Limited Data
 - better model ⇒ more training data
 - ⇒ small validation ⇒ noisy estimation of generalization ability

Measuring Generalization Ability

- 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
- Information Criteria
 - Pros
 - rely only on training set
 - need only (each) single training process to compare multiple models

2.2.2 Over-/Under-fitting

Over-fitting

- Symptom
 - good performance on training set & poor generalization
 - ⇒ 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)

Under-fitting

- bad at fitting training examples & modeling underlying data source
- lack of representation ability
- \Rightarrow poor performance on training set & good generalization (though meaningless)

2.3 Decision Theory

2.3.1

2.4 Information Theory

2.5 Supervised Learning

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

2.6 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
 - $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, $\mathbf{t} = \{t_1, \dots, t_N\}$
- Log Likelihood
 - $\ln P(\mathbf{y}|\mathbf{X}, \theta, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} \sum_{i=1}^m (h_\theta(x^i) - y^i)^2$
- 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 :

$$\Rightarrow \ln p(\theta|X) \propto -\frac{\beta}{2} \sum_{i=1}^n (y^i - h_\theta(x))^2 - \frac{\alpha}{2} \theta^T \theta + \text{const}$$
 - If $\alpha \rightarrow 0$ (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 Gaussian 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 Loss = $(\text{bias})^2 + \text{variance} + \text{noise}$
- Notation:
 - $t = y(x, w) + \epsilon$, where ϵ is Gaussian noise
 - \hat{y} is prediction function to approximate $y = y(x, w)$
- Procedure:
 - $\mathbb{E}[(t - \hat{y})^2] = \mathbb{E}[t^2 - 2t\hat{y} + \hat{y}^2]$
 $= \mathbb{E}[t^2] + \mathbb{E}[\hat{y}^2] - \mathbb{E}[2t\hat{y}]$
 $= \text{Var}[t] + \mathbb{E}[t]^2 + \text{Var}[\hat{y}] + \mathbb{E}[\hat{y}]^2 - 2y\mathbb{E}[\hat{y}]$
 $= \text{Var}[t] + \text{Var}[\hat{y}] + (y^2 - 2y\mathbb{E}[\hat{y}] + \mathbb{E}[\hat{y}]^2)$
 $= \text{Var}[t] + \text{Var}[\hat{y}] + (y - \mathbb{E}[\hat{y}])^2$
 $= \text{Var}[t] + \text{Var}[\hat{y}] + \mathbb{E}[t - \hat{y}]^2$
 $= \sigma^2 + \text{Var}[\hat{y}] + \text{Bias}[\hat{y}]^2$
 where $\sigma^2 = \text{Var}[\epsilon]$ is the noise
 (formula used: $\text{Var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 \Leftrightarrow \mathbb{E}[x^2] = \text{Var}[x] + \mathbb{E}[x]^2$)
 - Matrix inverse can be evil & avoid inverse operation:
 $A = U\Lambda U^T$, where Λ is diagonal matrix
 $\Rightarrow A^{-1} = U\Lambda^{-1}U^T$
 but number on the diagonal line of Λ can be small \Rightarrow maybe 0 depending on accuracy of computer

2.7 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 : $p(\mathbf{t}|w) = \prod_{n=1}^N \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\Phi w, \beta^{-1}I)$
- \Rightarrow Posterior : $p(w|\mathbf{t}) = \mathcal{N}(w|m_N, S_N)$
where $m_N = S_N(S_0^{-1}m_0 + \beta\Phi^T\mathbf{t})$, $S_N^{-1} = S_0^{-1} + \beta\Phi^T\Phi$
- Maximum Likelihood:
 - Likelihood : $p(\mathbf{t}|w) = \prod_{n=1}^N \mathcal{N}(t_n|\phi(x_n)w, \beta^{-1})$
 - meaning: how probable the observed dataset is w.r.t the model setting (under parameter w)
 - \ln Likelihood = $\sum_{n=1}^N [-\ln \frac{\beta}{\sqrt{2\pi}} - \frac{\beta}{2}(t_n - \phi(x)w)^2]$
 - $\frac{\partial}{\partial w} \ln$ Likelihood = $\beta\Phi^T(\mathbf{t} - \Phi w)$
let $\frac{\partial}{\partial w} \ln$ Likelihood = 0
 $\Rightarrow w_{ML} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{t}$
 - $\frac{\partial}{\partial \beta} \ln$ Likelihood = $-N\beta^{\frac{1}{2}} + \beta^{\frac{3}{2}}(\mathbf{t} - \Phi w)^T(\mathbf{t} - \Phi w)$
let $\frac{\partial}{\partial \beta} \ln$ Likelihood = 0
 $\Rightarrow \beta^{-1} = \frac{1}{N}(\mathbf{t} - \Phi w)^T(\mathbf{t} - \Phi w)$
Note: solve $w = w_{ML}$ first
- Maximum Posterior:
 - Posterior = $p(w|\mathbf{t})$, Prior = $p(w)$, Likelihood = $p(\mathbf{t}|w)$, Normalization = $p(\mathbf{t})$
 \Rightarrow Posterior = $\frac{\text{Likelihood} \cdot \text{Prior}}{\text{Normalization}}$
 \Rightarrow Posterior \propto Likelihood * Prior
 - assume Prior $p(w) = \mathcal{N}(w|m_0, S_0)$,
so that Prior & Likelihood are conjugate \Rightarrow Gaussian Posterior
 - Likelihood $p(\mathbf{t}|w) = \prod_{n=1}^N \mathcal{N}(t_n|\phi(x_n)w, \beta^{-1}) = \mathcal{N}(\mathbf{t}|\Phi w, \beta^{-1}I)$
 - \Rightarrow Posterior $p(w|\mathbf{t}) = \mathcal{N}(w|m_N, S_N)$,
where $m_N = S_N(S_0^{-1}m_0 + \beta\Phi^T\mathbf{t})$, $S_N^{-1} = S_0^{-1} + \beta\Phi^T\Phi$
 $\Rightarrow w_{MAP}$ = 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 $p(w|\mathbf{t}) = \mathcal{N}(w|m_N, S_N)$,

where $m_N = \beta(\alpha I + \beta\Phi^T\Phi)^{-1}\Phi^T\mathbf{t}$, $S_N^{-1} = \alpha I + \beta\Phi^T\Phi$

$w_{MAP} \rightarrow w_{ML}$, when $\alpha \rightarrow 0$ (most useless Prior)

- Maximize Posterior \Leftrightarrow Minimize cost function with regularization:

Simple Prior $\Rightarrow \ln p(w|\mathbf{t}) = -\frac{\beta}{2}(\mathbf{t} - \Phi w)^T(\mathbf{t} - \Phi w) - \frac{\alpha}{2}w^T w + \text{const}$

- If $\alpha \rightarrow 0$ (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 : $p(x|\alpha) = \mathcal{N}(x|0, \alpha^{-1}I)$, ($m_0 = 0, S_0 = \alpha^{-1}I$)

- $p(t|x, X, \mathbf{t}) = \int p(t|w, x)p(w|X, \mathbf{t})dw$

- $\Rightarrow p(t|x, X, \mathbf{t}) = \mathcal{N}(t|m_N^T\phi(x), \sigma_N^2(x))$

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 finding the Posterior

- Gradient descent

- Hypothesis function:

$$\blacksquare h_\theta(x) = x\theta, \theta = [\theta_0, \theta_1, \dots, \theta_n]^T, x = [x_0, x_1, \dots, x_n], x_0 = 1$$

- x is one instance

- Cost function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^i) - y^i)^2 + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$

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

$$\frac{d}{d\theta} J(\theta) = \frac{1}{m} ((X\theta - y)^T X)^T + \frac{\lambda}{m} [0, \theta_1, \dots, \theta_m]^T \quad (\theta_0 \text{ shouldn't be penalized})$$

- simultaneously for all $\theta_j \in \theta$

- Normal equation (mathematical solution)

$$\blacksquare \theta = (X^T X)^{-1} X^T y$$

2.8 Logistic Regression (Classification)

- Decision Theory:

- classes C_1, \dots, C_K , decision regions $\mathcal{R}_1, \dots, \mathcal{R}_K$ - Minimize $p(\text{mistake}) = \sum_{k=1}^K \left(\int_{\mathcal{R}_k} \sum_{i \neq k} p(x, C_i) dx \right)$

(can have weight on each misclassification though) - Maximize $p(\text{correct}) = \sum_{k=1}^K \int_{\mathcal{R}_k} p(x, C_k) dx$

- 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 training 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 $w = [w_1, \dots, w_n]^T$, $x = [x_1, \dots, x_n]$ - $y(x) > 0$:

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

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

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

- $y(x)$ 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 C_k if $\forall j \neq k, y_k(x) > y_j(x)$

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

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

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

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

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: $\tilde{m}_k = w^T m_k$, where $m_k = \frac{1}{N_k} \sum_{x \in C_k} x$ - Variance: $\tilde{s}_k = \sum_{x \in C_k} (w^T x - w^T m_k)^2 =$

$$w^T \left[\sum_{x \in C_k} (x - m_k)(x - m_k)^T \right] w$$

- 2-Classes to 1-D line:

- Maximize Fisher criterion: $J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$

- 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: $J(w) = \frac{\text{tr}\{W^T S_B W\}}{\text{tr}\{W^T S_W W\}}$

- Lagrangian:

Solve for each $w_i \in W \Rightarrow (S_W^{-1} S_B) w_i = \lambda_i w_i$

$\Rightarrow W$ consists 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 = $K - 1$:

$$r(S_W^{-1} S_B) \leq \min(r(S_W^{-1}), r(S_B)) \leq r(S_B)$$

$$r(S_B) \leq \sum_K r((m_k - m)(m_k - m)^T) = K, \text{ as } r(m_k - m) = 1$$

$$\sum_K m_k = m \Rightarrow r(m_1 - m, \dots, m_K - m) = K - 1$$

$$\Rightarrow r(S_B) \leq K - 1$$

$$\Rightarrow r(S_W^{-1} S_B) \leq K - 1$$

3. Perceptron Algorithm

- Generalised linear model $y = f(w^T \phi(x))$, where $\phi(x)$ is basis function; $\phi_0(x) = 1$

- Nonlinear activation function: $f(a) = \begin{cases} 1, & a \geq 0 \\ -1, & a < 0 \end{cases}$

- 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 training pair (x_n, t_n)

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

where $\eta=1$ 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 \mathbf{X}, \mathbf{t} , N is the num of data - $p(\mathbf{X}, \mathbf{t} | \pi, \mu_1, \mu_2, \Sigma) =$

$$\prod_{n=1}^N [\pi \mathcal{N}(x_n | \mu_1, \Sigma)]^{t_n} [(1 - \pi) \mathcal{N}(x_n | \mu_2, \Sigma)]^{1-t_n} \rightarrow \text{when info of label } t \text{ lost: mixture of Gaussian}$$

$$\ln(\text{Likelihood}) = \sum_{n=1}^N [t_n (\ln \pi + \ln \mathcal{N}(x_n | \mu_1, \Sigma)) + (1-t_n) (\ln(1-\pi) + \ln \mathcal{N}(x_n | \mu_2, \Sigma))] - \text{Parameters}$$

when maximum reached: - $\pi = \frac{N_1}{N_1 + N_2}$, N_1 is the num of class C_1 - $\mu_1 = \frac{1}{N_1} \sum_{n=1}^N t_n x_n, \mu_2 =$

$$\frac{1}{N_2} \sum_{n=1}^N (1-t_n) x_n, (\text{mean of each class}) - \Sigma = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2, \text{ where } S_k = \frac{1}{N_k} \sum_{n \in C_k} (x_n - \mu_k)(x_n - \mu_k)^T$$

5. Logistic Regression

- Sigmoid function: $\sigma(a) = \frac{1}{1 + e^{-a}}$

- $p(x|C_k) \sim \mathcal{N} \Rightarrow p(C_k|x) = \sigma(w^T x + w_0)$ (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

-

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

$$\text{where } a = \ln \frac{p(x, C_1)}{p(x, C_2)}$$

$$= \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$$

= ... (assumption applied)

$$= \ln \frac{\exp(\mu_1^T \Sigma^{-1} x - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1)}{\exp(\mu_2^T \Sigma^{-1} x - \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2)} + \ln \frac{p(C_1)}{p(C_2)}$$

$$\Rightarrow a = w^T x + w_0 \text{ where,}$$

$$w = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2 + \ln \frac{p(C_1)}{p(C_2)}$$

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

⇒ Find parameters in Gaussian model using Maximal Likelihood Solution

as: $p(C_1|x) \propto p(x|C_1)p(C_1) = p(x, C_1)$

- Generalize to K-classes:

$$a_k(x) = \ln[p(x|C_k)p(C_k)], p(C_k|x) = \frac{\exp(a_k)}{\sum_i \exp(a_i)}$$

$$\Rightarrow a_k(x) = w_k^T x + w_{k0}, \text{ where } w_k = \Sigma^{-1} \mu_k; w_{k0} = -\frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + p(C_k)$$

- Assume directly $p(C_k|x) = \sigma(w^T x + w_0)$ (2-classes) (Discriminative model)

- Assume directly: $p(C_1|w, x) = \sigma(w^T x), x_0 = 1$

⇒ less parameters to fit (compared to Gaussian)

- Likelihood function:

$$p(\mathbf{t}|w, X) = \prod_{n=1}^N p_n^{t_n} (1 - p_n)^{1-t_n}, \text{ where, } p_n = p(C_1|x_n), t_n \text{ is the class of } x_n$$

Define error function :

$$E(w) = -\ln(\text{Likelihood}) = -\sum_{n=1}^N [t_n \ln p_n + (1 - t_n) \ln(1 - p_n)]$$

$$\Rightarrow \nabla E(w) = \sum_{n=1}^N (p_n - t_n) x_n$$

- Find Posterior $p(w|\mathbf{t})$:

Likelihood is product of sigmoid

Conjugate Prior for "sigmoid distribution" is unknown

⇒ Assume Prior $p(w) = \mathcal{N}(w|m_0, S_0)$

$$\Rightarrow \ln p(w|\mathbf{t}) \propto -\frac{1}{2} (w - m_0)^T S_0^{-1} (w - m_0) + \sum_{n=1}^N [t_n \ln p_n + (1 - t_n) \ln(1 - p_n)]$$

$$\text{find } w_{MAP}, \text{ calculate } S_N = -\nabla \nabla \ln p(w|\mathbf{t}) = S_0^{-1} + \sum_{n=1}^N p_n (1 - p_n) \phi_n \phi_n^T$$

⇒ $p(w|\mathbf{t}) \simeq \mathcal{N}(w|w_{MAP}, S_N)$, via Laplace Approximation

- Laplace Approximation:

- Fit a gaussian 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: $f(z) \simeq f(z_0) \exp -\frac{A}{2} (z - z_0)^2$

⇒ Laplace Approximation = $(\frac{A}{2\pi})^{1/2} \exp -\frac{A}{2} (z - z_0)^2$, where $A = \frac{1}{\sigma^2}$

- 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 $A = -\nabla \nabla \ln f(z)|_{z=z_0}$

$$\Rightarrow \text{Laplace approximation} = \frac{|A|^{1/2}}{(2\pi^{M/2})} \exp -\frac{1}{2} (z - z_0)^T A (z - z_0) \quad (2.1)$$

$$= \mathcal{N}(z|z_0, A^{-1}) \quad (2.2)$$

- Gradient descent:

- Hypothesis function: $h_\theta(x) = \sigma(x\theta) = \frac{1}{1+e^{-x\theta}}$

- Cost function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y^i \ln(h_\theta(x^i)) - (1 - y^i) \ln(1 - h_\theta(x^i))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

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

2.9 Diagnose Machine Learning

- Size of data set - value of λ - Num of feature - Degree of polynomial - Build a quick implementation

2.9.1 Evaluating Hypothesis

1. Data set = training set (60)

- randomly split

2. Cross Validation:

- estimation of the generalization error, inaccurate because:

- finite training set - finite cross validation set

- S-fold Cross Validation:

- divide 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 training set

run the procedure (mentioned later) on the training set

estimate generalization in CV set

3. Test set error

- Linear regression: $J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^i) - y_{test}^i)^2$ - Logistic regression: $J_{test}(\theta) =$

$$-\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} (y_{test}^i \log h_\theta(x_{test}^i) + (1 - y_{test}^i) \log h_\theta(x_{test}^i)) - \text{Misclassification error: } J_{test}(\theta) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \text{err}(h_\theta(x_{test}^i), y_{test}^i)$$

$$\text{err}(h_\theta(x), y) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \text{ and } y = 0 \text{ or } h_\theta(x) < 0.5 \text{ and } y = 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.3)$$

4. Choosing procedure:

- Minimize training error $J_{train}(\theta)$ - Select a model with lowest $J_{cv}(\theta)$ - Estimate generalization error as $J_{test}(\theta)$

2.9.2 Bias / Variance

1. Problems:

- High bias: underfitting - high $J_{train}(\theta)$ and high $J_{cv}(\theta)$ - High variance: overfitting - low $J_{train}(\theta)$ but high $J_{cv}(\theta)$

2. Interaction with regularization:

- Improper λ : - large λ = high bias - small λ = high variance - Choosing λ : - try

$\lambda = 0, 0.01, 0.02, 0.04, \dots, 10$ - 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

- 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:

- $U = (u_1, \dots, u_M)$, where $\forall i \in \{1, \dots, M\}, u_i \in \mathbb{R}^D$ s.t. $u_i^T u_i = 1$ (only consider direction)

- Projected Data:

- Mean = $\bar{x}^T U$, where $\bar{x} = \frac{1}{N} \sum_{i=1}^N x^i$ - Variance = $\text{tr}\{U^T S U\}$, where $S = \sum_{i=1}^N (x^i - \bar{x})(x^i - \bar{x})^T$ (outer product)

- Lagrangian to maximize Variance:

- $L(U, \lambda) = \text{tr}\{U^T S U\} + \text{tr}\{(I - U^T U)\lambda\}$

constraint $u_i^T u_i = 1$ to prevent $u_i \rightarrow +\infty$

$$\text{For each } u_i \in U, \frac{\partial}{\partial u_i} L = 2S u_i - 2\lambda_i u_i = 0 \quad (2.4)$$

$$\Rightarrow S u_i = \lambda_i u_i \quad (2.5)$$

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

(S symmetric $\Rightarrow U$ orthogonal)

5. Minimum Error Formulation:

- Introduce Orthogonal Basis Vector for D dimension:

- $U = (u_1, \dots, u_D)$

- 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, \dots, z_M^n) is different for different x^n , (b_{M+1}, \dots, b_D) is the same for all x^n

- Cost function: $J = \frac{1}{N} \sum_{n=1}^N \|x^n - \widetilde{x}^n\|^2$, where $\widetilde{x}^n = \sum_{i=1}^M z_i^n u_i + \sum_{i=M+1}^D b_i u_i$

$$\text{- Let } \begin{cases} \frac{\partial}{\partial z_j^n} J = 0 \\ \frac{\partial}{\partial b_j} J = 0 \end{cases} \Rightarrow \begin{cases} \frac{1}{N} 2(x^n - \widetilde{x}^n)^T (-u_j) = \frac{2}{N} (z_j - (x^n)^T u_j) = 0 \\ \frac{1}{N} \sum_{n=1}^N 2(x^n - \widetilde{x}^n)^T (-u_j) = \frac{2}{N} \sum_{n=1}^N (b_j - (x^n)^T u_j) = 0 \end{cases}$$

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

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

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

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

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

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

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

$$(2.11)$$

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

- Lagrangian to Minimize J :

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

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

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

$$\Rightarrow S u_i = \lambda_i u_i$$

\Rightarrow To minimize 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

-

$$\text{Intuition: } \widetilde{x}_n = \sum_{i=1}^M ((x^n)^T u_i) u_i + \sum_{i=M+1}^D (\bar{x}^T u_i) u_i \quad (2.12)$$

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

1. Singular Value Decomposition - SVD:

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

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

eigenvalue decomposition: (2.14)

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

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

- Let $r(A) = r$

$$\Rightarrow r(A^T A) = r(A) = r \quad (2.16)$$

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

- Reduce AV to basis (Av_1, \dots, Av_r)

- Let $U = (u_1, \dots, u_r) = \left(\frac{Av_1}{\sqrt{\lambda_1}}, \dots, \frac{Av_r}{\sqrt{\lambda_r}} \right)$, 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: $\|u_i\| = \frac{\|Av_i\|}{\sqrt{\lambda_i}} = \frac{\sqrt{\langle Av_i, Av_i \rangle}}{\sqrt{\lambda_i}} = 1$

$\Rightarrow U$ is standard orthogonal (orthonormal) basis

- $AV = U\Sigma$, where $\Sigma = D^{\frac{1}{2}}$

- Expand U to orthonormal in $\mathbb{R}^m : (u_i, \dots, u_m)$

- Expand 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 covariance 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:

- $\tilde{X} = XV_M$, 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 $n \times k$ matrix, z is $k \times 1$ vector - [Reconstruction from data Compression](../Machine

4. Choosing k (num of principal components):

- choose the ****smallest**** k making $\frac{J}{V} \leq 0.01 = 1 - 99$

- $[U, S, V] = \text{svd}(\text{Sigma}) = 1 - \frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}}$, 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 $X^T - \forall n, y_n = D^{-\frac{1}{2}} V^T (x^n - \bar{x})$, where \bar{x} is the mean of X

$\Rightarrow y^n$ has zero mean (2.18)

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

6. Tips for PCA:

- Do NOT use PCA to prevent overfitting, use regularization instead - Try original data before implement PCA - Train PCA only on training 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 - Ignore 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 local distances in the original data

2.10.4 Anomaly Detection

1. Problem to solve:

- Given dataset x^1, x^2, \dots, x^m , build density estimation model $p(x) - p(x^{test} < \epsilon) = \epsilon$ 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, \dots, x_n independent from each other

3. Multivariate Gaussian:

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

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

$$- \mu = \frac{1}{m} \sum_{i=1}^m x_i, \Sigma = \frac{1}{m} \sum_{i=1}^m (x_i - \mu)(x_i - \mu)^T - x_1, \dots, x_n \text{ can be correlated but **not** linearly}$$

dependent - need $m > n$ ($m \geq 10n$ suggested) or else 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-gaussian 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 separate linear regression problem with the feature vectors of its rated items as training set

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

=> given X estimate Θ

- Cost Function for θ_j :

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

- Cost Function for Θ :

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

$\theta^j \in R^{n+1}$ (θ_0 not regularized), n_u is num of users

- Update Rule: $\forall \theta_k^j \in \theta^j, \theta_k^j := \theta_k^j - \alpha \frac{\partial J(\Theta)}{\partial \theta_k^j}, \frac{\partial J(\Theta)}{\partial \theta_k^j} = \sum_{i:r_{i,j}=1} (x^i \theta^j - y_{i,j}) x_k^i + \lambda \theta_k^j$, 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 : $J(x^i) = \frac{1}{2} \sum_{j:r_{i,j}=1} (x^i \theta^j - y_{i,j})^2 + \frac{\lambda}{2} \sum_{k=1}^n (x_k^i)^2$ - Cost Function for X :

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

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

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

- Basic Idea:

- Randomly initialize Θ

- loop:

Estimate X

Estimate Θ

- Cost Function:

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

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

- Update Rule:

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

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

- **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, \dots, 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 small => item i and j is similar

- Mean Normalization:

- Problem: if user j hasn't rated any movie, $\theta^j = [0, \dots, 0]$

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

=> useless prediction
 - Algorithm (row version):
 compute vector $\mu, \forall \mu_i \in \mu, \mu_i = \text{mean of } Y_i$, where Y_i is the i^{th} row in Y
 manipulate Y : $\forall y_{i,j} \in Y \wedge r_{i,j} = 1, y_{i,j} - \mu_i = \tilde{y}_{i,j}$ 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 descent

- Cost Function: - Cost function on single data: $\text{cost}(\theta, (x^i, y^i)) = \frac{1}{2}(h_\theta(x^i) - y^i)^2$ - Overall

Cost Function: $J_{\text{train}}(\theta) = \frac{1}{m} \sum_{i=1}^m \text{cost}(\theta, (x^i, y^i))$

- Procedure:
 - Randomly shuffle dataset
 - Repeat
 for $i \in [1, m]$

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

$$(\text{for } j = 0, \dots, n) \quad (2.21)$$

$$\Rightarrow \text{make progress with each single data} \quad (2.22)$$

- Convergence:
 - Wanting θ to converge => slowly decrease α over time (but more parameters)
 (E.g $\alpha = \frac{\text{const}_1}{\text{iteration num} + \text{const}_2}$)
 - Compute $\text{cost}(\theta, (x^i, y^i))$ before updating
 For every k update iterations, plot average $\text{cost}(\theta, (x^i, y^i))$ 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 in 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 Stochastic Gradient Descent: - 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 descent 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

- Character 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: Introduce distortions to the original data set - Need to identify the appropriate distortion - Usually adding purely/random/meaningless 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 - Calculate the time it needs before decide to/not to collect the data

2.12.3 Ceiling 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)

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
 - Learning 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)
 - $\text{ReLU} \Leftrightarrow$ 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)
 - \Rightarrow 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 \rightarrow coordinates \rightarrow linear transform \rightarrow 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... \Rightarrow single general algorithms to learn

12.1.3 Research Advices

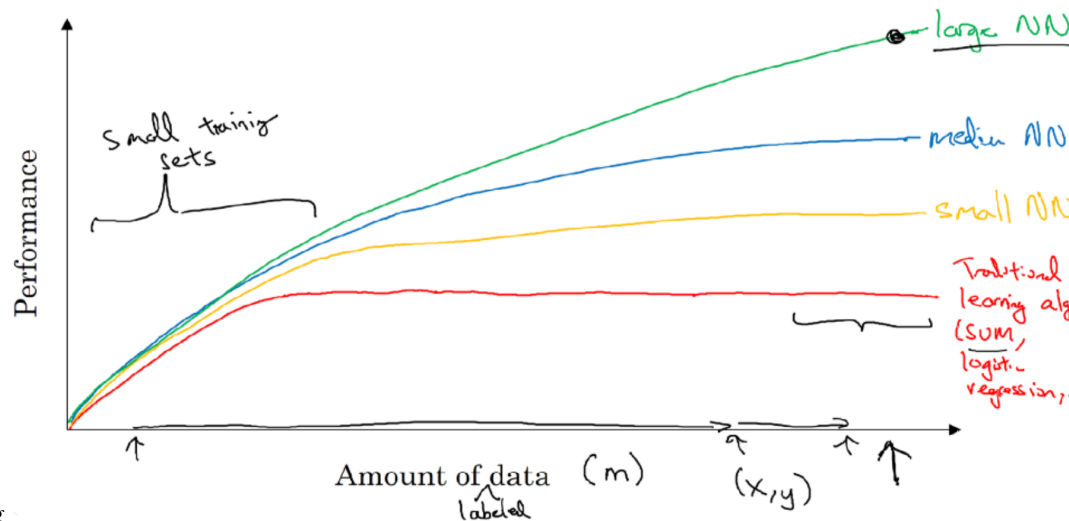
- Reading
 - read a little bit & find somewhere intuitively not right
 - good intuition: eventually work; bad intuition: not working no matter what it is doing
 - if other doubts your idea as bullshit \Rightarrow a sign for real good result
 - a supervisor with similar belief
 - PhD vs. Company
 - amount of mentoring
 - faster if dedicated supervisor available
- Practice
 - open-source learning resource
 - implement the paper

12.2 Basic Neural Network

12.2.1 Goal and Advantages

Data Mining and Pattern Recognition

- Larger Maximum Capability
 - Curve given Amount of Data



Learning/background-largedata.png

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

Ability to Choose

- Choosing 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

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

- Ability of Depth (however)
 - 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
- Possible Solutions
 - Xavier initialization, ReLU activation, ...

12.2.3 Learning

Forward-Backward Propagation

- Representation
 - Layers
 - input layer, hidden layer(s) (each has a bias term $x_0 = 1$)
 - 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_j^l : activation of unit j at layer l
 - $h(\cdot)$: activation function (usually shared)
 - z_j^l : output of unit j at layer l (represent parameterized basis)
- Forward Propagation
 - Activation $a^{j+1} = w^j \cdot [z_0^j, \dots, z_{s_j}^j]^T$, with $z_0 = 1$
 - Unit Output $z^{j+1} = h(a^{j+1}) = [z_1^{j+1}, \dots, z_{s_{j+1}}^{j+1}]^T$
- Backward Propagation
 -

12.3 Advanced Topic

12.3.1 Machine Reading Comprehension

Problem Formulation

-

RNN with Attention

Convolution with Self-attention - QAnet

12.3.2 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 $t = 1, \dots, T$
 - LSTM scanning the sentence, with hidden state h_t at time t
 - language info $l_t = \text{concat}[h_t, w_t]$
 - Combining Info
 - l_t tiled to spatial info, at each time step
 \Rightarrow creating combined feature maps F_t (of shape [height, wide, channel])
 - combined feature maps F_1, \dots, 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
 ⇒ an 1-D global encoding for each time step (via weighted mean over space)
 ⇒ filling feature maps: global encoding if attention here > threshold; else **0**
 ⇒ summing filled feature maps over time for the global spatial maps c
 - attention weighting over tiled language info at each time step, correspondingly
 ⇒ 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

Research Direction

- Integrating Encoder-Decoder Architecture
 - Upsampling
- Early Info Combination
 - Tiling at First Conv
 - downsampling more responsible for language info processing
 ⇒ hopefully get more fine-tuning + more non-linearity for language info
 - can used with pre-trained net:

$$ReLU(conv_1 * X_1 + conv_2 * X_2) = ReLU([conv_1, conv_2] * [X_1, X_2])$$

- Multiple Entry
 - 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)
⇒ auxiliary loss
 - Language as Conv Filter
 - Language Info, through a subnet, becoming a set of conv filters
⇒ then imposed in downsampling, tunnel, upsampling or bridge stage(s)