Chapter 2

Machine Learning Fundamentals

Qiang Ji

Table of Contents

Machine Learning

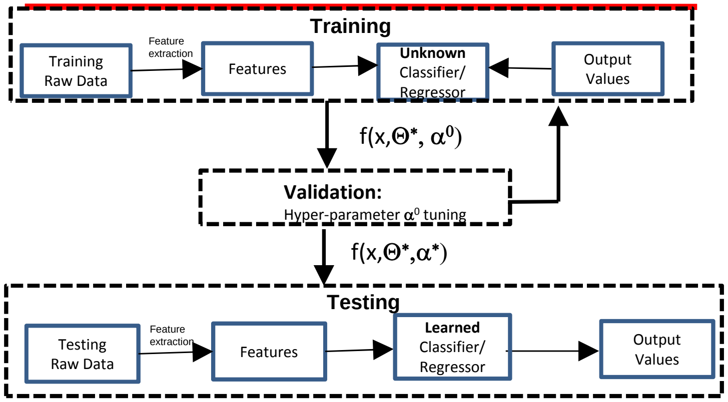
$$\mathbf{x} \longrightarrow f(\mathbf{x},\Theta,\alpha) \longrightarrow \mathbf{y}$$

- **x**-input, **y**-output
- f() the mapping function that maps input to output
- Θ and α are the parameters and hyper-parameters of the mapping function

Types of Learning

- Supervised learning
  - Training data includes both inputs and the corresponding outputs
- Unsupervised learning
  - Training contains only inputs and does not include desired output
- Reinforcement learning
  - Inputs include data and rewards
  - Outputs include actions to take

Supervised Learning



Basic mathematics

- Probability calculus
- Linear algebra
- Multivariable calculus
- Gradient based learning

Only provide minimally sufficient materials.
Read related references for details.

Probability Calculus

- **Random variable**: a random variable (RV) is a variable whose value is uncertain, depending on its chance as a result of a random process that maps a RV into a specific value.
- Let capital X represent a RV and lower case x ∈ χ represent a particular value of X, where χ *defines the value space*
- *A RV can be discrete or continuous. A discrete RV assumes a finite set of values while a continuous RV assumes a real value.*

Probability Calculus

- *The chance of a RV assuming a particular value is measured by its probability, i.e.,*

  P(X=x) or p(x) in short , and $0 \le p(x) \le 1$
- *P(X) represents the probability distribution (pdf) of X and*
  - *For discrete RV,*

    $$\sum_{x \in \chi} p(x) = 1$$
  - For continuous RV,

    $$\int_{x \in \chi} p(x) dx = 1$$

Probability Calculus (cont'd)

- A **random vector** consists of a vector of RVs. We use bold variable to represent a random vector, i.e., $\mathbf{X}=(X_1, X_2, \ldots, X_N)^t$, and we use a lower case bold **x** to represent a value vector of **X,** i.e., **X**=**x.** Both **X** and **x** are column vectors.
- p(**X**) represents the probability of the random vector, i.e., the joint probability of all RVs in **X.**

Expectation (Mean)

- For a discrete RV X
  $$E(X) = \sum_{x \in X} x \bullet p(x)$$
- For a continuous RV X
  $$E(X) = \int_x x \bullet p(x)\,dx$$
- Conditional Expectation
  $$E(X \mid y) = \int_x x \bullet p(x \mid y)\,dx$$
- For a random vector $\mathbf{X}=(X_1, X_2, \ldots, X_N)^t$
  $E(\mathbf{X})=(E(X_1), E(X_2), \ldots, E(X_N))^t$

Variance

- The variance of a RV X
  $$Var(X) = \int [X - E(X)]^2 p(x)\,dx$$
  $$= E[(X - E(X))^2] = E(X^2) - E^2(X)$$
- Standard deviation
  $$\sigma_X = \sqrt{Var(X)}$$
- $Var(X|y)=E[(x-E(X|y))^2]=E(X^2|y)-E^2(X|y)$

Covariance and Covariance Matrix

- Covariance of RVs X and Y
  $$\sigma^2{}_{XY} = E[(X - E(X))(Y - E(Y))]$$
  $$= E(XY) - E(X)E(Y)$$
- Covariance Matrix-variance of a random vector $X=(X_1, X_2, \ldots, X_N)^t$
  $$\Sigma_{\mathbf{X}}^{NxN} = E[(\mathbf{X} - E(\mathbf{X}))(\mathbf{X} - E(\mathbf{X}))^t]$$

Diagonal-variance Off-diagonal-covariance

Probability Rules

- Product rule
- Union rule
- Chain rule
- Sum rule
- Conditional probability rule
- Bayes' rule

Product Rule

Given two RVs X and Y, let p(X,Y) represent their joint probability and p(X|Y) represent the conditional probability of X given Y,

$$p(X,Y)=p(X|Y)p(Y)$$

$$p(X \mid Y) = \frac{p(X,Y)}{p(Y)}$$

Union Rule

Given two RVs X and Y, let p(X+Y) represent the probability of their union, i.e., the probability of X or Y.

$$P(X+Y)=p(X)+p(Y)-p(X,Y)$$

if X and Y are mutually exclusive, $p(X,Y)=0$

$$P(X+Y)=p(X)+p(Y)$$

Chain Rule

Given three RVs A, B, C ,

Chain rule :

$$p(A,B,C) = p(A)\,p(B \mid A)\,p(C \mid A,B)$$

Conditional chain rule:

$$p(A,B,C \mid D,E) = p(A \mid D,E)\,p(B \mid A,D,E)\,p(C \mid A,B,D,E)$$

Any assumptions? Chain rule for N variables X1, X2, …, X_N?

Sum Rule

- Sum rule via marginalization

$$P(X) = \sum_Y P(X,Y) = \sum_Y \sum_Z P(X,Y,Z)$$

- Sum rule via conditional probability

$$P(X) = \sum_Y P(X \mid Y)\, p(Y)$$

$$P(X \mid Y) = \sum_Z P(X \mid Y, Z)\, p(Z \mid Y)$$

Bayes' Rule

prior probability of X

Likelihood of X

$$p(X \mid Y) = \frac{p(X)\, p(Y \mid X)}{p(Y)}$$

p(Y) is a normalization constant $p(Y) = \sum_X p(X)\, p(Y \mid X)$

Independence

- If X and Y are marginally independent, then

$$p(X,Y) = p(X)\,p(Y)$$
$$p(X \mid Y) = ?$$
$$E(XY) = ?$$
$$Cov(X,Y) = ?$$

We denote it as $X \perp Y$

Conditional Independence

- For three RVs, X, Y, and Z, if X and Y are independent, given Z , we have

$$P(X \mid Y, Z) = P(X \mid Z)$$
$$p(X, Y \mid Z) = ?$$
$$E(X, Y \mid Z) = ?$$

We denote it as

$$X \perp Y \mid Z$$

Probability Distributions

- Probability distribution can be discrete or continuous.
- Probability distribution for one RV, i.e., univariate distribution or probability distribution for multiple RVs, i.e., multivariate or joint distribution

Discrete Probability Distributions

- Uniform distribution X∈ $\{1,2,\ldots,K\}$

$$X \sim Uniform(x \mid K) \quad P(X = k \mid K) = \frac{1}{K}$$

- Bernoulli distribution for a binary RV X∈ $\{0,1\}$

$$X \sim Ber(x \mid \theta),\ P(X = 1) = \theta,\ p(X = 0) = 1 - \theta$$

- Binominal distribution
  - Let Y be an integer RV that represents the number of times for X=1 for N Bernoulli trials, with $p(X = 1) = \theta$

$$Y \sim Bin(y \mid N, \theta) \quad P(Y = n_1 \mid N, \theta) = \binom{N}{n_1} \theta^{n_1} (1-\theta)^{N-n_1}$$

Discrete Probability Distributions

- Multinomial distribution for K discrete RVs, Y_1, Y_2, …, Y_K, where Y_k is an integer that represents the number of times a discrete RV X∈ $\{1,2,\ldots,K\}$ equal to k out of a total of N trials of X. Let **Y**=(Y_1,Y_2,…,Y_K)

$$Y \sim Mult(y_1, y_2, \ldots, y_k \mid N, \theta_1, \theta_2, \ldots, \theta_K) = \frac{N!}{y_1! y_2! \ldots y_k!} \theta_1^{y_1} \theta_2^{y_2} \ldots \theta_k^{y_K}$$

where $p(X = k) = \theta_k$

Binominal distribution is a special case of multinomial distribution, where K=2

Continuous Probability Distributions

- Gaussian distribution
  - Unary: $p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
  - Multivariate $\mathbf{X} \sim N(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = p(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} e^{-\frac{(\mathbf{x}-\boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}{2}}$
- Dirichlet distribution
  - Continuous multivariate probability distributions for K RVs $X_1,X_2,\ldots,X_K$, where $X_i \in (0,1)$ and $\sum_{i=1}^{K} X_i = 1$

$$\mathbf{X} \sim Dir(x_1, x_2, \ldots, x_K \mid \alpha_1, \alpha_2, \ldots, \alpha_K) = p(x_1, x_2, \ldots, x_K \mid \alpha_1, \alpha_2, \ldots, \alpha_K) = \frac{1}{B(\alpha_1, \alpha_2, \ldots, \alpha_k)} \prod_{i=1}^{K} x_i^{\alpha_i - 1},$$

$$where\ B(\alpha_1, \alpha_2, \ldots, \alpha_K) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k} \alpha_i)} \quad and \quad \Gamma(n) = (n-1)!$$ gamma function

Linear Algebra

- A scalar is a single number. It can be real ($\mathbb{R}$), integer ($\mathbb{N}$), boolean, binary, etc. . It is represent by a lower case symbol: x, n, i, j

  $x \in \mathbb{R}$ – a real scalar $x \in \mathbb{N}$ – an integer scalar

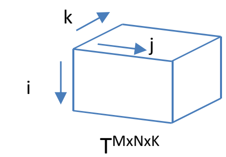
- A vector is a 1-D array of scalars, represented by a bold upper case letter in column vector

  $$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{pmatrix} \qquad \mathbf{x} \in \mathbb{R}^N$$ a vector of real numbers

Linear Algebra

- A matrix **A** (i,j) is a 2D array of scalars, represented by a uppercase bold letter

$$\mathbf{A}^{MxN} = \begin{pmatrix} a_{11}\, a_{12} \dots & a_{11} & a_{1N} \\ a_{21}\, a_{22} \dots & a_{21} & a_{2N} \\ \dots & & \\ a_{M1}\, a_{M2} \dots a_{M1} & & a_{MN} \end{pmatrix} \qquad \mathbf{A} \in R^{MxN}$$



- A tensor T is a N-D dimensional array of scalars **T**(i,j, k,..). A tensor becomes a scalar, vector, and matrix when N=0, 1, and 2. When N=3, tensor T is a 3D matrix **T**(i,j,k).

Vector Norms

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_1 \\ \dots \\ x_N \end{pmatrix} \qquad |\mathbf{X}|_p = (\sum_{i=1}^{N} |x_i|^p)^{\frac{1}{p}}$$

$$L_1 : |\mathbf{X}|_1 = \sum_{i=1}^{N} |x_i|$$

$$L_2 : |\mathbf{X}|_2 = (\sum_{i=1}^{N} x_i^2)^{\frac{1}{2}} = \sqrt{x_1^2 + x_2^2 + \dots + x_N^2}, \text{ the Euclidean distance}$$

$$L_\infty : |\mathbf{X}|_\infty = \max_i |x_i|$$

$$L_0 = \#(i \mid x_i \neq 0) - \text{the cardinality of } \mathbf{X}$$

Matrix Norms

$$\mathbf{A}^{MxN} = \begin{pmatrix} a_{11}\, a_{12} \dots a_{1N} \\ a_{21}\, a_{22} \dots a_{2N} \\ \dots \\ a_{M1} a_{M2} \dots a_{MN} \end{pmatrix}$$

$$L_1 : |\mathbf{A}|_1 = \max_{1 \le j \le M} \sum_{i=1}^{N} |a_{ij}| = \max(\sum_{i=1}^{N} |a_{i1}|, \sum_{i=1}^{N} |a_{i2}|, \dots, \sum_{i=1}^{N} |a_{iM}|)$$

$$L_2 : |\mathbf{A}|_2 = (\sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij}^2)^{\frac{1}{2}}$$

Trace and Determinant

- Trace of matrix
  - Sum of the diagonal elements, i.e.,

$$Tr(\mathbf{A}) = \sum_i a_{i,i}$$

- Determinant of A
  - Product of all eigenvalues of A , i.e.,
- Rank of A
  - The number of independent rows or columns

Eigenvalues and Eigenvectors

- An eignvector of a square matrix A^{NxN} is a non-zero vector v such that multiplication of **A** by **v** alters only the scale of **v**, i.e.,

$$\mathbf{Av} = \lambda \mathbf{v}$$

where λ is called the eignvalue of A.

- A has N eigenvetors ($\mathbf{v}_n$)and N eigenvalues (λ_n)

Eigen-decomposition

- Given its eigenvalues λ_n and eigenvectors $\mathbf{v}_n$ of a matrix, a matrix **A** can be decomposed into

$$\mathbf{A} = V\Lambda V^{-1} \qquad V^{NxN} = [\mathbf{v}_1\, \mathbf{v}_2 \ldots \mathbf{v}_N] \qquad \Lambda^{NxN} = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \; \lambda_N \end{bmatrix}$$

Note **V** is orthonormal, i.e., $\mathbf{V}^{-1}=\mathbf{V}^t$ if A is real and symmetric.

Singular Value Decomposition (SVD)

- A matrix $\mathbf{A}^{MxN}$ can be decomposed into the products of three matrices

$$\mathbf{A}^{MxN} = \mathbf{U}^{MxM}\mathbf{D}^{MxN}\mathbf{V}^{t\,NxN}$$

where U are V are orthonormal matrices ($\mathbf{U}^{-1}=\mathbf{U}^t$) and D is a diagonal matrix, whose values are called singular values

SVD v.s. Eigen-decomposition

- For a square symmetric matrix $\mathbf{A}^{MxM,}$ its SVD is

$$\mathbf{A} = \mathbf{UD}\, U^t$$

- Its eigen decomposition is

$$\mathbf{A} = V \Lambda V^{-1}$$

- Hence, U=V and D =Λ

SVD v.s. Eigen-decomposition

- For a non-square matrix $\mathbf{A}^{MxN,}$ its SVD is

$$\mathbf{A} = \mathbf{UD}\ \mathbf{V}^t$$

- The columns of **V** are eigenvectors of $\mathbf{A}^t\mathbf{A}$.
- The columns of **U** are eigenvectors $\mathbf{A}\mathbf{A}^t$
- The non-zero singular values of **D** are the square roots of the nonzero eigenvalues of $\mathbf{A}\mathbf{A}^t$ (or $\mathbf{A}^t\mathbf{A}$)

SVD Applications

- The rank of **A** is the number of non-zero singular values.
- Computing matrix inverse

$$\mathbf{A}^{-1} = (\mathbf{UD}\ \mathbf{V}^t)^{-1} = \mathbf{VD}^{-1}\mathbf{U}^t$$

- Computing eigen vectors
  - For a symmetric matrix **A**, its SVD decomposition

$$\mathbf{A} = V\mathbf{D}\ \mathbf{V}^t$$

  - The columns of **V** are eigenvectors of **A** and the singular values in **D** are the eigenvalues of **A**

System of Linear Equations

- Let $\mathbf{X}^{Nx1}$ be a unknown vector, $\mathbf{A}^{MxN}$ and $\mathbf{b}^{Mx1}$ are given. Find **x** by minimizing

$$(\mathbf{Ax}\text{-}\mathbf{b})^t(\mathbf{Ax}\text{-}\mathbf{b})$$

- If M≥N and the rows are independent, **x** has an unique solution

$$\mathbf{x} = (A^t A)^{-1} A^t \mathbf{b}$$

where $(A^t A)^{-1} A^t$ is called pseudo-inverse of **A** can be computed via SVD

- If M < N, **x** has multiple solutions

Principal Component Analysis (PCA)

- Given data **D**=[**x**[m]],m=1,2,..,M, compute the covariance matrix Σ for samples in **D**
- Perform SVD on $\Sigma=UDV^t$, and obtain the eigen vectors of from columns of V matrix
- Order the eigen vectors in a descending order according to their eigen values
- Select top K eigen vectors and form a project matrix M, whose rows are the top K selected eigen vectors
- Multiply each input **x** by M to produce the projected data y,i.e., y=Mx

Multivariable (Vector) Calculus

- Involves differentiation and integration of vectors and matrices
- Derivatives with vector
  - Scalar by vector
  - Vector by scalar
  - Vector by vector
- Derivatives with Matrices
  - Scalar by matrix
  - Vector by matrix
- Follow denominator layout convention

Derivatives with Vectors

Let $\mathbf{X}^{Nx1}$ and $\mathbf{Y}^{Mx1}$ be column vectors and *z* be a scalar function of z(**X**), following the denominator layout

- Scalar by vector

$$\frac{\partial z}{\partial \mathbf{X}}^{Nx1} = \begin{pmatrix} \frac{\partial z}{\partial X_1} \\ \frac{\partial z}{\partial X_2} \\ \dots \\ \frac{\partial z}{\partial X_N} \end{pmatrix}$$

- Vector by scalar

$$\frac{\partial \mathbf{X}^{1xN}}{\partial z} = \begin{pmatrix} \frac{\partial X_1}{\partial z} & \frac{\partial X_2}{\partial z} \dots \frac{\partial X_N}{\partial z} \end{pmatrix}$$

Vector by Vector

- Vector by vector

$$\left(\frac{\partial \mathbf{X}^{Nx1}}{\partial \mathbf{Y}^{Mx1}}\right)^{MxN} = \begin{pmatrix} \frac{\partial \mathbf{X}_1}{\partial \mathbf{Y}} & \frac{\partial \mathbf{X}_2}{\partial \mathbf{Y}} \dots \frac{\partial \mathbf{X}_N}{\partial \mathbf{Y}} \end{pmatrix} = \begin{pmatrix} \frac{\partial X_1}{\partial Y_1} & \frac{\partial X_2}{\partial Y_1} \dots \frac{\partial X_N}{\partial Y_1} \\ \frac{\partial X_1}{\partial Y_2} & \frac{\partial X_2}{\partial Y_2} \dots \frac{\partial X_N}{\partial Y_2} \\ \dots & \\ \frac{\partial X_1}{\partial Y_M} & \frac{\partial X_2}{\partial Y_M} \dots \frac{\partial X_N}{\partial Y_M} \end{pmatrix}$$

- Let $\mathbf{V}^{Mx1}$ be column vector function $\mathbf{X}^{Nx1}$ and A^{KxM} be a matrix that is NOT a function of **X**

$$\frac{\partial (AV)}{\partial X}^{NxK} = \frac{\partial V}{\partial X} A^t$$

Scalar Function by Vector

Let $\mathbf{U}^{Mx1}(\mathbf{X})$ and $\mathbf{V}^{Mx1}(\mathbf{X})$ be column vectors, and both are a function $\mathbf{X}^{Nx1}$.

$$\frac{\partial (U^t V)}{\partial X}^{Nx1} = \frac{\partial U}{\partial X} V + \frac{\partial V}{\partial X} U$$

Scalar Function by Vector

Let $\mathbf{U}^{Kx1}$ and $\mathbf{V}^{Mx1}$ be column vectors, and both are a function $\mathbf{X}^{Nx1}$, A^{KxM} be a matrix that is NOT a function of $\mathbf{X}$

$$\frac{\partial (U^t A V)}{\partial X}^{Nx1} = \frac{\partial V}{\partial X} A^t U + \frac{\partial A^t U}{\partial X} = \frac{\partial V}{\partial X} A^t U + \frac{\partial U}{\partial X} A V$$

Scalar Function by Vector

Given A^{MxN}, b^{Mx1}, C^{MxK}, D^{KxN}, e^{Kx1} that are Not a function $\mathbf{X}^{Nx1}$,

$$\frac{\partial (AX+b)^t C(DX+e)}{\partial X} = \frac{\partial (DX+e)}{\partial X} C^t (AX+b) + \frac{\partial (AX+b)}{\partial X} C(DX+e)$$
$$= D^t C^t (AX+b) + A^t C(DX+e)$$

Derivatives with Matrices

Let $\mathbf{A}^{MxN}$ be a matrix and *z* be a scalar function of A

- Scalar by matrix

$$\left(\frac{\partial z^{1x1}}{\partial \mathbf{A}^{MxN}}\right)^{MxN} = \left(\frac{\partial z}{\partial \mathbf{A}[\,][1]} \; \frac{\partial z}{\partial [\,][2]} \cdots \frac{\partial z}{\partial [\,][N]}\right) = \begin{pmatrix} \frac{\partial z}{\partial A[1][\,]} \\ \frac{\partial z}{\partial A[2][\,]}. \\ \cdots \\ \frac{\partial z}{\partial A[M][\,]} \end{pmatrix} = \begin{pmatrix} \frac{\partial z}{\partial a_{11}} \; \frac{\partial z}{\partial a_{12}} \cdots \frac{\partial z}{\partial a_{1N}} \\ \frac{\partial z}{\partial a_{21}} \; \frac{\partial z}{\partial 2_{22}} \cdots \frac{\partial z}{\partial a_{2N}} \\ \cdots \\ \frac{\partial z}{\partial a_{M1}} \; \frac{\partial z}{\partial a_{M2}} \cdots \frac{\partial z}{\partial a_{MN}} \end{pmatrix}$$

where A[][i] – ith column of A, A[i][-]- ith row of A

- Let $\mathbf{u}^{Mx1}$ and $\mathbf{v}^{Nx1}$ are not function of A

$$\frac{\partial \mathbf{u}^t \mathbf{A} \mathbf{v}}{\partial A}^{MxN} = \mathbf{u}\mathbf{v}^t$$

Derivatives with Matrices

- Vector by matrix-Tensor . Let $\mathbf{X}^{Kx1}$ be a vector

$$\left(\frac{\partial \mathbf{X}^{Kx1}}{\partial \mathbf{A}^{MxN}}\right)^{MxNxK} = \left(\frac{\partial \mathbf{X}}{\partial \mathbf{A}[][1]} \frac{\partial \mathbf{X}}{\partial \mathbf{A}[][2]} \cdots \frac{\partial \mathbf{X}}{\partial \mathbf{A}[][N]}\right) = \left(\frac{\partial \mathbf{X}[1]}{\partial \mathbf{A}} \frac{\partial \mathbf{X}[2]}{\partial \mathbf{A}} \cdots \frac{\partial \mathbf{X}[k]}{\partial \mathbf{A}}\right)$$

$$= \begin{pmatrix} \frac{\partial \mathbf{X}}{\partial \mathbf{A}[1][]} \\ \frac{\partial \mathbf{X}}{\partial \mathbf{A}[2][]} \\ \cdots \\ \frac{\partial \mathbf{X}}{\partial \mathbf{A}[M][]} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{X}[1]}{\partial \mathbf{A}[1][]} & \frac{\partial \mathbf{X}[2]}{\partial \mathbf{A}[1][]} & \cdots & \frac{\partial \mathbf{X}[K]}{\partial \mathbf{A}[1][]} \\ \frac{\partial \mathbf{X}[1]}{\partial \mathbf{A}[2][]} & \frac{\partial \mathbf{X}[2]}{\partial \mathbf{A}[2][]} & \cdots & \frac{\partial \mathbf{X}[K]}{\partial \mathbf{A}[2][]} \\ \cdots & & & \\ \frac{\partial \mathbf{X}[1]}{\partial \mathbf{A}[M][]} & \frac{\partial \mathbf{X}[2]}{\partial \mathbf{A}[M][]} & \cdots & \frac{\partial \mathbf{X}[K]}{\partial \mathbf{A}[M][]} \end{pmatrix}$$

Derivatives with Matrices

- Tensor vector multiplication

  $Let\ \ \mathbf{Y}\ \text{be a Kx1 vector}$

$$\frac{\partial \mathbf{X}}{\partial \mathbf{A}}^{MxNxK} \mathbf{Y}^{Kx1} = \left(\sum_{k=1}^{K} \frac{\partial \mathrm{X}_k}{\partial \mathbf{A}} \ \mathrm{Y}_k\right)^{MxN}$$

- Tensor matrix multiplication

  $Let\ \ \mathbf{B}\ \text{be a KxD matrix}$

$$\frac{\partial \mathbf{X}}{\partial \mathbf{A}}^{MxNxK} \mathbf{B}^{KxD} = \sum_{k=1}^{K} \frac{\partial \mathrm{X}_k}{\partial \mathbf{A}} B[k][]$$

Derivatives with Matrices

- Derivative of Determinant of A

$$\frac{\partial |\mathbf{A}|}{\partial \mathbf{A}} = |A|\, A^{-t}$$

Parameter Learning

$$\mathbf{x} \longrightarrow \mathrm{f}(\mathbf{x},\Theta) \longrightarrow \mathbf{y}$$

- Parameter learning is to learn the mapping function parameters **Θ**, given M i.i.d training data **D**={X[m],Y[m]}, where m=1,2,..,M
- Parameter learning is often formulated as finding **Θ** to minimize certain loss function L(**D**:Θ).

Parameter Learning (cont'd)

- Let l(X[m],y[m]) be the loss function for each sample, L(**D**:Θ) is defined below
  - Unregularized loss function (average loss)
    - L(**D**:Θ) $=\frac{1}{M}\sum_{m=1}^{M} l(X[m], y[m], \boldsymbol{\Theta})$,
  - Regularized loss function
    - L(**D**:Θ) $==\frac{1}{M}\sum_{m=1}^{M} l(X[m], y[m], \boldsymbol{\Theta})$+λR(Θ), R(**Θ**) is a regularization term
- Given the loss function, **Θ** is solved by minimizing the loss function, i.e.,

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} L(\mathbf{D} : \boldsymbol{\Theta})$$

Gradient-based Parameter Learning

For some loss function, analytic closed form solution exists. Parameters **Θ** can be solved by computing the gradient of the loss function with respect to **Θ** and setting it to zero

$$\nabla_{\Theta} L(\boldsymbol{D}:\Theta) = \frac{\partial L(\mathbf{D} : \boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} = 0$$

Gradient based Parameter Learning

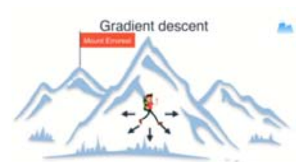
- For many problems, analytic solution may not exist, gradient descent is often used to iteratively estimate Θ

$$\boldsymbol{\Theta}^{t} = \boldsymbol{\Theta}^{t-1} - \eta_t \frac{\partial L(\mathbf{D} : \boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta} = \boldsymbol{\Theta}^{t-1}}$$

  where Θ^t is the estimate at t th iteration, η is the learning rate that needs be manually tuned. **Θ** needs be initialized to initial value Θ^0
- Gradient descent method is the dominant learning technique for deep learning

Gradient Descent

- Descent the mountain iteratively. At each iteration, the best direction to descend is the negative of the gradient
- For convex loss function, descent will converge to global minimum independent of initialization.
- For non-convex loss function, descent may stuck in a local minimum
- In practice, local minimum does not seem to be a problem-a theoretical mystery!



Gradient based Parameter Learning

- For non-regularized loss function, gradient of the parameters can be computed as follows

$$\frac{\partial L(\mathbf{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}} = \frac{1}{M}\sum_{m=1}^{M}\frac{\partial l(\mathbf{x}[m], y[m], \mathbf{\Theta})}{\partial \mathbf{\Theta}}$$

- For regularized loss function,

$$\frac{\partial L(\mathbf{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}} = \frac{1}{M}\sum_{m=1}^{M}\frac{\partial l(\mathbf{x}[m], y[m], \mathbf{\Theta})}{\partial \mathbf{\Theta}} + \lambda\frac{\partial R(\mathbf{\Theta})}{\partial \mathbf{\Theta}}$$

Stochastic Gradient Descent (SGD)

- When the number of samples (M) is large, computing the gradient of Θ using all samples can be very slow.

$$\frac{\partial L(\mathbf{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}} = \frac{1}{M}\sum_{m=1}^{M}\frac{\partial l(\mathbf{x}[m], y[m], \mathbf{\Theta})}{\partial \mathbf{\Theta}}$$

- Stochastic Gradient Descent (SGD) can approximately compute the gradient

Stochastic Gradient Descent

Mini-batch approximation

- Divide **D** into K batches-D_1, D_2, …D_K with the batch size S ranging from 2 to 100.
- At each iteration t, randomly select a batch D_k and compute the gradient

$$\frac{\partial L(\mathbf{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}} = \frac{1}{S}\sum_{\mathbf{x}[m], y[m]\in D_k}\frac{\partial l(\mathbf{x}[m], y[m], \mathbf{\Theta})}{\partial \mathbf{\Theta}}$$

$$\mathbf{\Theta}^{t} = \mathbf{\Theta}^{t-1} - \eta_t \left.\frac{\partial L(\mathbf{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}}\right|_{\mathbf{\Theta}=\mathbf{\Theta}^{t-1}}$$

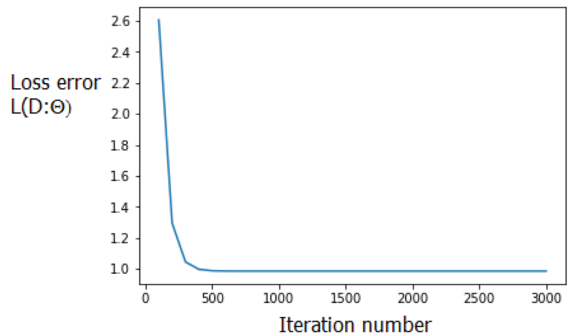
- The size of the batch (a hyper-parameter) is independent of the number of samples

Convergence of GD/SGD

- The gradient descent method iterates from an initial point until convergence
- The convergence can be measured by the change of
  - the estimated parameters Θ or
  - the magnitude of the gradient $|\nabla L(D:\Theta)|$ or
  - the loss function value

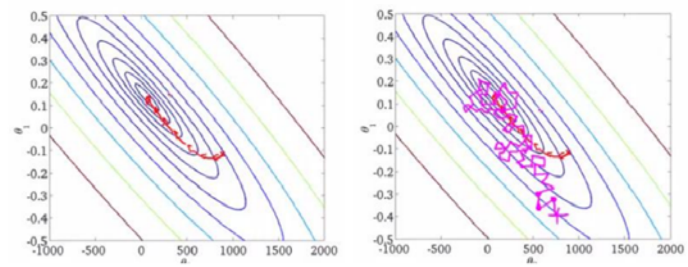
  When the change is below a threshold, the iteration can stop.
- The iteration can also be stopped when the maximum iteration number is reached.

Convergence of GD/SGD



GD versus SGD

- Batch gradient descent vs. Stochastic Gradient Descent



Source: https://www.cs.cmu.edu/~yuxiangw/docs/SSGD.pdf

Regression

The goal of regression is to predict the value of one or more continuous target output y (the *regress and or dependent variable)* values given the value of an input feature vector **x**

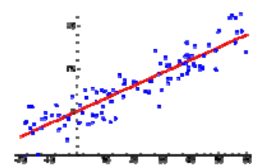
$$f: \mathbf{x} \in R^N \rightarrow y \in R^1$$

Linear Regression

- Linear regression means the output y is a linear function of the parameters **w** of the regression model, i.e.,

$$y=\mathbf{w}^t\mathbf{x}+w_0$$

where **w** is a vector of weight coefficients and w_0 a bias



Linear Regression (cont'd)

$$y(\mathbf{x},\mathbf{w}) = \mathbf{w}^t\mathbf{x} + w_0 = [\mathbf{x}^t \quad 1]\begin{bmatrix}\mathbf{w}\\ w_0\end{bmatrix} = \mathbf{X}^t\mathbf{\Theta}$$

where $\mathbf{x}^{N\times1}=(x_1,x_2,\ldots,x_N)^t$ is input vector, y (a real scalar), the output variable, and $\mathbf{w}=\{w_1,w_2,\ldots,w_N\}$ is the model parameters, and w_0 is the bias.

Given $\mathbf{\Theta} = \begin{bmatrix}\mathbf{w}\\ w_0\end{bmatrix}$, linear regression is to predict the value of y for a new input **x**

Linear Regression Learning

Given paired training data **D**={**x**[m], y[m]}, m=1,2,…,M, learn the parameters $\mathbf{\Theta}$ of the regression function f(**x**,Θ) by minimizing some loss functions, i.e.,

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} L(\mathrm{D}:\mathbf{\Theta})$$

where L() is a loss function. Types of loss functions:

1) Mean squared errors (MSE)

$$l(x[m],y[m]) = (\boldsymbol{x}^t[\boldsymbol{m}]\boldsymbol{w} + w_0 - y[m])^2$$

2) Negative Log Conditional Likelihood (-LCL) (also called cross-entropy)

$$l(x[m],y[m]) = -logp(y[m]|\boldsymbol{x}[m],\boldsymbol{\theta})$$

Mean Squared Errors

Let $\mathbf{\Theta} = \begin{bmatrix}\mathbf{w}\\ w_0\end{bmatrix}$ be the model parameters

and $\mathbf{X} = \begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix}$ be the homogeneous representation of the input

$$L_{MSE}(\mathrm{D}:\mathbf{\Theta}) = \frac{1}{M}\sum_{m=1}^{M}(\mathbf{w}^t\mathbf{x}[m] + w_0 - y[m])^2$$

$$= \frac{1}{M}\sum_{m=1}^{M}(\mathbf{X}^t[m]\mathbf{\Theta} - y[m])^2$$

$$= \frac{1}{M}(\underbrace{\begin{bmatrix}\mathbf{X}^t[1]\\ \mathbf{X}^t[2]\\ \ldots\\ \mathbf{X}^t[M]\end{bmatrix}}_{\mathbf{A}}\mathbf{\Theta} - \underbrace{\begin{bmatrix}y[1]\\ y[2]\\ \ldots\\ y[M]\end{bmatrix}}_{\mathbf{Y}})^t(\begin{bmatrix}\mathbf{X}^t[1]\\ \mathbf{X}^t[2]\\ \ldots\\ \mathbf{X}^t[M]\end{bmatrix}\mathbf{\Theta} - \begin{bmatrix}y[1]\\ y[2]\\ \ldots\\ y[M]\end{bmatrix})$$

$$= \frac{1}{M}(\mathbf{A\Theta} - \mathbf{Y})^t(\mathbf{A\Theta} - \mathbf{Y})$$

Least-squares Estimation

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} L_{MSE}(\mathrm{D}:\mathbf{\Theta})$$

$$\frac{\partial L_{MSE}(\mathrm{D}:\mathbf{\Theta})}{\partial\mathbf{\Theta}} = 0$$

$$\mathbf{\Theta}^* = (\mathrm{A}^t\mathrm{A})^{-1}\mathrm{A}^t\mathbf{Y}$$

Negative Log Conditional Likelihood

$$y = \mathbf{w}^t\mathbf{x} + w_0 + \varsigma$$
$$\varsigma \sim N(0, \sigma^2)$$
$$p(y \mid \mathbf{x}) \sim N(\mathbf{w}^t\mathbf{x} + w_0, \sigma^2)$$
$$L_{-LCL}(D:\boldsymbol{\Theta}) = -\sum_{m=1}^{M} \log p(y[m] \mid \mathbf{x}[m]) \qquad \text{cross-entropy}$$
$$= -\sum_{m=1}^{M} \log\{\frac{1}{\sqrt{(2\pi)}\sigma} e^{-\frac{(y[m]-\mathbf{w}^t\mathbf{x}[m]-w_0)^2}{2\sigma^2}}\}$$
$$= \sum_{m=1}^{M} -\frac{(y[m]-\mathbf{w}^t\mathbf{x}[m]-w_0)^2}{2\sigma^2} + M\log\sigma + \frac{M}{2}\log(2\pi)$$
$$\approx \frac{(\mathbf{A\Theta}-\mathbf{Y})^t(\mathbf{A\Theta}-\mathbf{Y})}{2\sigma^2} + M\log\sigma$$

Negative Log Conditional Likelihood

$$L_{-LCL}(D:\boldsymbol{\Theta}) = \frac{(\mathbf{A\Theta}-\mathbf{Y})^t(\mathbf{A\Theta}-\mathbf{Y})}{2\sigma^2} + M\log\sigma$$

- It is clear that the mean squared loss function is a special case of the negative log conditional likelihood function with σ treated as constant
- σ is an additional parameter that can also be learnt

Maximum Likelihood Estimation

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} L_{-LCL}(D:\boldsymbol{\Theta})$$
$$= \arg\max_{\boldsymbol{\Theta}} LCL(D:\boldsymbol{\Theta})$$
$$\frac{\partial L_{-LCL}(D:\boldsymbol{\Theta})}{\partial\boldsymbol{\Theta}} = 0 \Rightarrow$$
$$\boldsymbol{\Theta}^* = (\mathrm{A}^t\mathrm{A})^{-1}\mathrm{A}^t\mathbf{Y}$$
$$\sigma^2 = \frac{(\mathbf{A\Theta}^* - \mathbf{Y})^t(\mathbf{A\Theta}^* - \mathbf{Y})}{M}$$

Learning with Regularization

- To control over-fitting, a regularization term is added to the loss function, creating a new target function

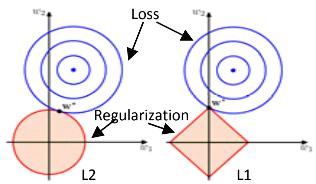
$$L(\boldsymbol{\Theta}:\mathbf{D}) + \lambda R(\boldsymbol{\Theta})$$

Loss function + Regularization term

- R(Θ) is typically squared L2 or L1 norm
  - Squared L2 norm $R(\Theta) = |\Theta|^2_2 = \Theta^t\Theta$
  - L1 norm $R(\Theta) = |\Theta|_1$
- λ is a hyper-parameter that determines the relative weight of the two terms

L2 versus L1 norm

- L1 normal (lasso) tends to generate sparser solutions by forcing parameters to become zero. It is convex but non-differentiable at zero.
- L2 norm (quadratic norm) produces small value (close to zero) parameters. It is differentiable and the regularized function remains convex. It is widely used.



Regression Learning with L1 Norm

- With a MSE loss function and a quadratic regularizer g(Θ)=$\Theta^T\Theta$, the objective function becomes

$$L_{MSEL2}(\mathrm{D}:\mathbf{\Theta}) = (\mathrm{A}\mathbf{\Theta}-\mathbf{Y})^t(\mathrm{A}\mathbf{\Theta}-\mathbf{Y}) + \lambda\mathbf{\Theta}^t\mathbf{\Theta}$$

$$\frac{\partial L_{MSEL2}(\mathrm{D}:\mathbf{\Theta})}{\partial\mathbf{\Theta}} = 2A^t(\mathrm{A}\mathbf{\Theta}-\mathbf{Y}) + 2\lambda\mathbf{\Theta} = 0$$

$$\mathbf{\Theta} = (A^tA+\lambda I)^{-1}A^tY$$

Note I is an identity matrix

Regression Learning with L2 Norm

- With a –LCL loss function and a quadratic regularizer g(Θ)=$\Theta^T\Theta$, the objective function becomes

$$L_{-LCLL2}(\mathrm{D}:\mathbf{\Theta}) = \frac{(\mathrm{A}\mathbf{\Theta}-\mathbf{Y})^t(\mathrm{A}\mathbf{\Theta}-\mathbf{Y})}{2\sigma^2} + M\log\sigma + \lambda\mathbf{\Theta}^t\mathbf{\Theta}$$

$$\frac{\partial L_{-LCLL2}(\mathrm{D}:\mathbf{\Theta})}{\partial\mathbf{\Theta}} = \frac{A^t(\mathrm{A}\mathbf{\Theta}-\mathbf{Y})}{\sigma^2} + 2\lambda\mathbf{\Theta} = 0$$

$$\mathbf{\Theta} = (A^tA+2\lambda\sigma^2MI)^{-1}A^tY$$

- $2\lambda\sigma^2$ can be treated as one composite hyper-parameter

Regression Learning with L1 Regularization

- With a MSE loss function and a L1 norm R(Θ)=$|\Theta|_1$, the objective function becomes

$$L_{MSEL1}(\boldsymbol{D}:\boldsymbol{\Theta}) = (A\boldsymbol{\Theta}-\boldsymbol{Y})^t\,(A\boldsymbol{\Theta}-\boldsymbol{Y}) + \lambda|\boldsymbol{\Theta}|_1$$

$$= (A\boldsymbol{\Theta}-\boldsymbol{Y})^t\,(A\boldsymbol{\Theta}-\boldsymbol{Y}) + \lambda\sum_{i=1}^{N}|\Theta_i|$$

Regression Learning with L1 Regularization

$$\frac{\partial MSEL1(\mathrm{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}} = 2A^t(\mathrm{A}\mathbf{\Theta}-\mathbf{Y}) + \lambda \frac{\partial |\mathbf{\Theta}|_1}{\partial \mathbf{\Theta}}$$

$$= 2A^t(\mathrm{A}\mathbf{\Theta}-\mathbf{Y}) + \lambda \begin{pmatrix} \frac{\partial |\mathbf{\Theta}|_1}{\partial \Theta_1} \\ \frac{\partial |\mathbf{\Theta}|_1}{\partial \Theta_2} \\ .... \\ \frac{\partial |\mathbf{\Theta}|_1}{\partial \Theta_N} \end{pmatrix} = 2A^t(\mathrm{A}\mathbf{\Theta}-\mathbf{Y}) + \lambda \begin{pmatrix} \mathrm{sign}(\Theta_1) \\ \mathrm{sign}(\Theta_2) \\ .... \\ \mathrm{sign}(\Theta_N) \end{pmatrix}$$

$$\mathrm{sign}(\Theta_i) = \begin{cases} 1 & \text{if } \Theta_i > 0 \\ -1 & \text{if } \Theta_i < 0 \\ c \in [-1,+1] \text{ if } & \Theta_i = 0,\ c \text{ is sometimes choen to be zero} \end{cases}$$

Hence, $\frac{\partial |\mathbf{\Theta}|_1}{\partial \mathbf{\Theta}}$ equals 2^{N} vectors of $+1$s or -1s, and it is nondifferentiable at $\Theta_i = 0$

Subgradient descent method can be used.