

# 0 Basic Math Fundamentals

## 0.1 Linear Algebra

### Norms:

$$\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}} \quad \text{for } p \geq 1$$
$$\|x\|_0 = \text{"number of non-zero entries of } x\text{"}$$

### Inverse/Rank:

$$(A_1 A_2 \dots A_k)^{-1} = A_k^{-1} \dots A_2^{-1} A_1^{-1} \quad \text{if } A_1, A_2, \dots, A_k \text{ invertible}$$
$$\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$$
$$\text{rank}(A^T A) = \text{rank}(A)$$

### Matrix-Sum Notation:

$$x^T A y = \sum_{i=1}^n x_i (A x)_i = \sum_{i=1}^n x_i \left( \sum_{j=1}^n A_{ij} y_j \right)$$
$$= \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i y_j = y^T A^T x$$

### Matrix multiplication:

$$C = AB, A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times p}, c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

### Positive and Negative Definiteness:

$$\text{PD: } x^T A x > 0 \quad \text{PSD: } x^T A x \geq 0$$

$$\text{ND: } x^T A x < 0 \quad \text{NSD: } x^T A x \leq 0$$

### Eigenvalues and -vectors: $A \in \mathbb{R}^{n \times n}$

$$A \text{ is PSD} \Leftrightarrow A \text{ has no negative eigenvalues}$$

$$A \text{ is PD} \Leftrightarrow A \text{ has only positive eigenvalues}$$

$$A \text{ is invertible} \Leftrightarrow A \text{ has only non-zero eigenvalues}$$

$$A \text{ symmetric} \Leftrightarrow \text{all } n \text{ eigenvec. of } A \text{ are orthonormal}$$

$$\text{rank}(A) = \# \text{ non-zero eigenvalues}$$

### Lagrangian:

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda g(x, y)$$

$$\nabla_{x,y,\lambda} \mathcal{L}(x, y, \lambda) = 0$$

### Derivatives:

$$\nabla_x x^T A x = (A + A^T) x \quad \nabla_x x^T a = \nabla_x a^T x = a$$

$$\nabla_x x^T x = \nabla_x \|x\|_2^2 = 2x \quad \frac{\partial A x}{\partial x} = A$$

$$\frac{\partial x^T A y}{\partial A} = x y^T \quad \frac{\partial x^T A^T y}{\partial A} = y x^T$$

$$\frac{\partial}{\partial A} (\log(\det A)) = (A^{-1})^T \quad \frac{\partial}{\partial A} \text{tr}(AB) = B^T$$

$$\frac{\partial}{\partial X} a^T X^{-1} b = -(X^{-1})^T a b^T (X^{-1})^T$$

$$\nabla_{x|s} (x - s)^T W (x - s) = -2W(x - s), W \text{ symmetric}$$

$$\nabla_x (Ax - s)^T (Ax - s) = 2(A^T A x - A^T s)$$

### Chain rule, etc.:

$$\frac{\partial v(x) a}{\partial x} = a \frac{\partial v(x)}{\partial x} \quad \frac{\partial v(x) u(x)}{\partial x} = v \frac{\partial u(x)}{\partial x} + u \frac{\partial v(x)}{\partial x}$$

$$\frac{\partial g(u)}{\partial x} = \frac{\partial g(u)}{\partial u} \frac{\partial u}{\partial x}$$

### Gradients:

$$\nabla_a c = \left( \frac{\partial c}{\partial a} \right)^T = \begin{bmatrix} \frac{\partial c}{\partial a_1} & \dots & \frac{\partial c}{\partial a_m} \end{bmatrix}^T \in \mathbb{R}^m$$

$$\left[ \frac{\partial a}{\partial W} \right]_{ijk} = \frac{\partial a_i}{\partial W_{jk}} \quad \text{with } a \in \mathbb{R}^H, W \in \mathbb{R}^{H \times D}, \frac{\partial a}{\partial W} \in \mathbb{R}^{H \times H \times D}$$
$$\frac{\partial W}{\partial c} \in \mathbb{R}^{H \times D}, \quad \frac{\partial c}{\partial W} \in \mathbb{R}^{D \times H}$$

### Jacobian:

$$J_a = \frac{\partial a}{\partial x} = \begin{bmatrix} \frac{\partial a_1}{\partial x_1} & \frac{\partial a_1}{\partial x_2} & \dots & \frac{\partial a_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_m}{\partial x_1} & \frac{\partial a_m}{\partial x_2} & \dots & \frac{\partial a_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$\text{Hessian: } \nabla_x^2 f(x) = J_{\nabla_x f(x)} = \frac{\partial}{\partial x} \nabla f(x) = \frac{\partial}{\partial x} \left( \frac{\partial f(x)}{\partial x} \right)^T$$
$$\nabla_x^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

## 0.2 Probability Theory

### 0.2.1 Basics

#### Independence & conditional independence:

$$P(A, B) = P(A)P(B)$$

$$P(A|B, C) = P(A|B) \Leftrightarrow P(A, B|C) = P(A|C)P(B|C)$$

#### Marginalization:

$$p(a) = \int \int p(a, b, c) db dc \quad p(a|x) = \sum_b p(a, b|x)$$

#### Mean, Variance and Covariance (Matrix):

$X, Y$ : 1D random variables;  $\mathbf{X}$ : random vector with  $n$  RV

$$\mathbb{E}[g(X)] = \sum_i g(x_i) p(x_i) = \int_{\mathbb{R}} g(x) p(x) dx$$

$$\mathbb{E}[g(X, Y)] = \sum_i \sum_j g(x_i, y_j) p(x_i, y_j) = \int_{\mathbb{R}} \int_{\mathbb{R}} g(x, y) p(x, y) dx dy$$

$$\mathbb{E}[g(\mathbf{X})] = \int_{\mathbb{R}^n} g(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}_1 \dots d\mathbf{x}_n$$

$$\text{Var}[f(X)] = \mathbb{E}[(f(X) - \mathbb{E}[f(X)])^2]$$
$$= \mathbb{E}[f(X)^2] - \mathbb{E}[f(X)]^2$$

$$\text{Cov}[X, Y] = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$
$$= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

$$\Sigma = \mathbb{E}[\mathbf{X}\mathbf{X}^T] - \mathbb{E}[\mathbf{X}]\mathbb{E}[\mathbf{X}]^T$$
$$= \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^T]$$

$$\mathbb{E}[aX + Y + b] = a \cdot \mathbb{E}[X] + \mathbb{E}[Y] + b$$

$$\text{Var}[aX + b] = a^2 \cdot \text{Var}[X]$$

$$\text{Var}[aX + bY] = a^2 \text{Var}[X] + b^2 \text{Var}[Y] + 2ab \text{Cov}[X, Y]$$

#### Chain Rule in Probability Theory:

$$P(A, B) = P(B|A) \cdot P(A) = P(A|B) \cdot P(B)$$

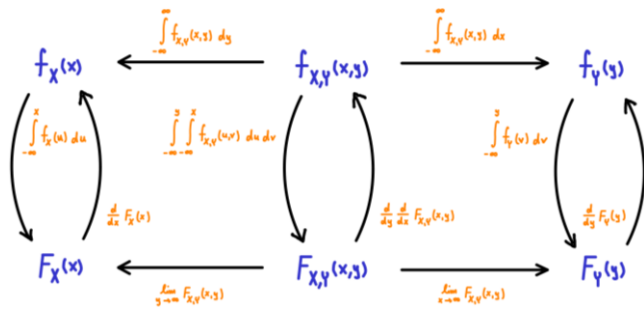
$$P(A, B|C) = \frac{P(A, B, C)}{P(C)} = P(A|B, C) \cdot P(B|C)$$

$$P(A, B, C) = P(C|A, B) \cdot P(A, B)$$

#### Bayes Theorem:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} = \frac{P(A, B)}{P(B)}$$

## PDF ↔ CDF:



## 0.2.2 Distributions

**Precision:**  $\beta = \frac{1}{\sigma^2}$ ,  $\binom{n}{k} = \frac{n!}{k!(n-k)!}$  ( $\triangleq$  nCr)

Distribution	PDF / PMF	Mean	Variance
Bernoulli( $x p$ )	$p^x(1-p)^{1-x} = \begin{cases} p, & \text{if } x = 1 \\ 1-p, & \text{if } x = 0 \end{cases}$	$p$	$p(1-p)$
Binomial( $k n, p$ )	$\binom{n}{k} p^k (1-p)^{n-k}$ ; $0 \leq k \leq n$	$np$	$npq$
Geometric( $k p$ )	$p(1-p)^{k-1}$ for $k = 1, 2, \dots$	$1/p$	$\frac{1-p}{p^2}$
Poisson( $x \lambda$ )	$e^{-\lambda} \lambda^x / x!$ for $k = 1, 2, \dots$	$\lambda$	$\lambda$
Uniform( $x a, b$ )	$\frac{1}{b-a} \forall x \in [a, b]$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
Gaussian $\mathcal{N}(x \mu, \sigma^2)$	$\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	$\mu$	$\sigma^2$
Laplace( $x \mu, b$ )	$\frac{1}{2b} \exp\left(-\frac{ x-\mu }{b}\right)$	$\mu$	$2b^2$
Exponential ( $x \lambda$ )	$\lambda e^{-\lambda x}$ ; $x \geq 0, \lambda > 0$	$1/\lambda$	$1/\lambda^2$
Beta( $\theta a, b$ )	$\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{a-1} (1-\theta)^{b-1}$ $\theta \in [0, 1], a > 0, b > 0$ $\Gamma(n) = (n-1)!$	$\frac{a}{a+b}$	$\frac{ab}{(a+b)^2(a+b+1)}$
Gamma( $x a, b$ )	$\frac{1}{\Gamma(a)} b^a x^{a-1} e^{-bx}$	$a/b$	$a/b^2$

Mode of Beta( $a, b$ ):  $\frac{a-1}{a+b-2}$ , for  $a, b > 1$

## 0.2.3 Gaussians

$p(x|\mu, \Sigma) = \mathcal{N}(x|\mu, \Sigma) \triangleq x \sim \mathcal{N}(\mu, \Sigma)$

**Multivariate Gaussian:**  $x \in \mathbb{R}^D$

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}} \det(\Sigma)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \\ \propto \exp\left(-\frac{1}{2}(x^T \Sigma^{-1} x - 2x^T \Sigma^{-1} \mu)\right)$$

**Log-space:**

$$\log \mathcal{N}(x|\mu, \Sigma) = -\frac{D}{2} \log 2\pi - \frac{1}{2} \log(\det \Sigma) - \frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)$$

$$\log \mathcal{N}(x|\mu, \sigma^2) = -\frac{1}{2} \log 2\pi - \frac{1}{2} \log(\sigma^2) - \frac{1}{2\sigma^2}(x-\mu)^2$$

**Convolution of Gaussians:**

For independent random variables  $X$  and  $Y$ , the distribution  $f_Z$  of  $Z = X + Y$  equals the convolution of  $f_X$  and  $f_Y$ :

$$f_Z(z) = f_X(z) * f_Y(z) = \int_{-\infty}^{\infty} f_Y(z-x) f_X(x) dx$$

For  $f_X(x) = \mathcal{N}(x|\mu_X, \sigma_X^2)$  and  $f_Y(y) = \mathcal{N}(y|\mu_Y, \sigma_Y^2)$ :

$$f_Z(z) = \int_{-\infty}^{\infty} f_Y(z-x) f_X(x) dx \\ = \int_{-\infty}^{\infty} \mathcal{N}(z-x|\mu_Y, \sigma_Y^2) \mathcal{N}(x|\mu_X, \sigma_X^2) dx \\ = \mathcal{N}(z|\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$$

**„Translation“ of Gaussians:**

$$p(x) = \mathcal{N}(x|\mu, \sigma^2) = \mathcal{N}(x-\mu|0, \sigma^2)$$

If  $y = x + a$  (for a fixed  $a \in \mathbb{R}$ ), then

$$p(y) = \mathcal{N}(y-a|\mu, \sigma^2) = \mathcal{N}(y|\mu+a, \sigma^2).$$

## 0.3 Tricks

**Determinant:**  $A \in \mathbb{R}^{n \times n}$ ,  $c \in \mathbb{R}$

$$\log(\det A) = -\log((\det A)^{-1}) = -\log(\det(A^{-1}))$$

$$\det(A^{-1}) = \frac{1}{\det(A)} \quad \det(c \cdot A) = c^n \cdot \det(A)$$

$$\log(\det(cA)) = \log(c^n \det(A)) = n \log(c) + \log(\det(A))$$

**Trace:**

$$\text{tr}(ABC) = \text{tr}(CAB) = \text{tr}(BCA), \quad ABC \in \mathbb{R}^{n \times n}$$

**Sum:**

$$\sum_i \sum_j x_i x_j = \sum_i x_i \cdot \sum_j x_j$$

**(Reverse) Triangle Inequality:**

$$\|x\| - \|y\| \leq \|x - y\|, \quad x, y \in \mathbb{R}^n$$

$$\|x + y\| \leq \|x\| + \|y\|$$

$$\|x\| - \|y\| \leq \|x - y\| \text{ (follows from reverse triangle inequality)}$$

**L<sub>2</sub>-norm trick:**

$$\|x\|_2^2 - \|y\|_2^2 = (x - y)^T (x + y), \quad x, y \in \mathbb{R}^n$$

**Law of iterated expectations:**

$$\mathbb{E}_X[X] = \mathbb{E}_Z[\mathbb{E}_X[X|Z]] = \sum_i \mathbb{E}_X[X|Z_i] \cdot P(Z_i)$$

**Exponential function:**

$$\prod_i \exp(x_i) = \exp(\sum_i x_i) \quad x_i \in \mathbb{R}$$

**Stacked vectors:**

$$\begin{pmatrix} a \\ b \end{pmatrix}^T \begin{pmatrix} c \\ d \end{pmatrix} = a^T c + b^T d, \quad a, c \in \mathbb{R}^n; b, d \in \mathbb{R}^m$$

**Rewrite binary cross entropy loss:**  $y_i \in \{0, 1\}$

$$E(z) = -\sum_{i=1}^N (y_i \log \sigma(z) + (1 - y_i) \log(1 - \sigma(z))) \\ = \sum_{i=1}^N (\ln(1 + \exp(z)) - y_i z)$$

## 0.4 Miscellaneous

**Local Maxima and Minima:**  $f: X \rightarrow \mathbb{R}$

If the domain  $X$  is a metric space, then  $f$  has a local maximum (minimum) point at the point  $x^*$ , if there exists some  $\varepsilon > 0$  such that  $f(x^*) \geq f(x)$  ( $f(x^*) \leq f(x)$ ) for all  $x$  in  $X$  within distance  $\varepsilon$  of  $x^*$ .

If  $g$  (e.g. exp) is a monotonic transform, we also have:

$$g(f(x^*)) \geq g(f(x)). \text{ Hence, } x^* \text{ is also a maximizer of } g \circ f.$$

**Logic:**

- Contraposition:  $(P \Rightarrow Q) \equiv (\neg Q \Rightarrow \neg P)$
- Decomposition:  $(X \perp\!\!\!\perp (A, B)) \Rightarrow (X \perp\!\!\!\perp A) \wedge (X \perp\!\!\!\perp B)$
- Contraction:  $(X \perp\!\!\!\perp A | B) \wedge (X \perp\!\!\!\perp B) \Rightarrow (X \perp\!\!\!\perp (A, B))$
- "A Concise Course in Statistical Inference" by L. Wasserman (Theorem 17.2):  
 $(X \perp\!\!\!\perp A | B) \wedge (X \perp\!\!\!\perp B | A) \Rightarrow (X \perp\!\!\!\perp (A, B))$

**Taylor expansion of the exponential function:**

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

**Geometric series:**

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x} \quad \text{for } |x| < 1$$

## 13 Advanced Topics

### Considerations:

Privacy, Security, Fairness, Explainability, Accountability

### 13.1 Differential Privacy (DP)

**Model Inversion** attacks recover information about the training data from the trained model.

**Randomized Response:** Introducing randomization to provide plausible deniability.

1. Flip a coin. If it lands tails answer truthfully.

2. Else flip another coin. If it lands tails, answer Yes, else No.

Unbiased Estimator of truthful Yes-answers:  $\mu = 2(\hat{\mu} - \frac{1}{4})$ ,

where  $\hat{\mu}$  is the MLE.  $\hat{\mu} = \frac{\# \text{ participants that said Yes}}{\# \text{ participants}}$

### Definition Differential Privacy:

A randomized mechanism  $\mathcal{M}_f: \mathcal{X} \rightarrow \mathcal{Y}$  is  $\epsilon$ -differentially private if **for all** neighboring inputs  $X \simeq X'$  and **for all** sets of outputs  $Y \subseteq \mathcal{Y}$  we have:

$$P(\mathcal{M}_f(X) \in Y) \leq e^\epsilon \cdot P(\mathcal{M}_f(X') \in Y)$$

For any possible set of outputs we have:

$$e^{-\epsilon} \leq \frac{P(\mathcal{M}_f(X) \in Y)}{P(\mathcal{M}_f(X') \in Y)} \leq e^\epsilon$$

*Intuition:* The outcome of the statistical analysis should not change by much if we include/exclude/modify the information of a single instance ( $\triangleq$  neighboring input).

$$P(\mathcal{M}_f(X) \in Y) = \int_Y p_X(y) dy \quad p_X(y): \text{PDF of } \mathcal{M}_f(X)$$

### Laplace mechanism (output perturbation):

- Define the global sensitivity of a function  $f: \mathcal{X} \rightarrow \mathbb{R}^d$  as  $\Delta_p = \sup_{X \simeq X'} \|f(X) - f(X')\|_p$
- $\Delta_p$  measures the magnitude by which a single instance can change the output of the function in the worst case

Output perturbation with Laplace noise:  $\mathcal{M}_{f,Lap}$  is  $\epsilon$ -DP

- Curator holds data  $X = (x_1, \dots, x_n) \in \mathcal{X}$  about  $n$  individuals and computes the function  $f(X)$
- Sample i.i.d. Laplace noise  $Z \sim \text{Lap}(0, \frac{\Delta_1}{\epsilon})^d$
- Reveal the noisy value  $\mathcal{M}_{f,Lap}(X) = f(X) + Z$
- $p_X(y) = \text{Lap}(y|f(X), \frac{\Delta_1}{\epsilon})^d = \prod_{i=1}^d \frac{\epsilon}{2\Delta_1} e^{-\frac{\epsilon}{\Delta_1}|y_i - f(X)_i|}$

**DP techniques for ML models:** Perturb input, weights, objective (loss function) or gradients.

### Fundamental properties of DP:

- Robustness to post-processing: If  $\mathcal{M}$  is  $\epsilon$ -DP, then  $g \circ \mathcal{M}$  is  $\epsilon$ -DP (as long as you don't touch again the data).
- Composition: If  $\mathcal{M}_j, j = 1, \dots, k$  are  $\epsilon_j$ -DP, then  $X \rightarrow (\mathcal{M}_1(X), \dots, \mathcal{M}_k(X))$  is  $(\sum_{j=1}^k \epsilon_j)$ -DP.
- Group privacy: If  $\mathcal{M}$  is  $\epsilon$ -DP with respect to  $X \simeq X'$ , then  $\mathcal{M}$  is  $(t\epsilon)$ -DP w.r.t changes of  $t$  instances/individuals.

**Federated Learning:** Learning a model without any centralized entity having access to all the data.  $\rightarrow$  Can be made DP.

## 13.2 Algorithmic Fairness

### Causes of bias:

Tainted training data, Skewed sample, Proxies (sensitive features may be highly correlated with other features), Sample size disparity, Limited features

### Notions of Fairness:

- Group fairness: Aims to treat all groups equally.
- Individual: Treat similar examples similar.
- Counterfactual: Uses tools from causal inference.

### Group fairness: Setup

$X \in \mathbb{R}^d$ : features of an individual,  $A \in \{a, b, \dots\}$ : sensitive features,  $R = r(X, A) \in \{0, 1\}$ : **binary predictor**,  $Y \in \{0, 1\}$ : ground truth/target,  $P_a(R) = P(R|A = a)$

### Fairness through Unawareness:

Exclude sensitive attributes from training data:  $R = r(X)$

+ Easy; Legal support | - Proxies of sensitive attributes included

### First criterion: Independence

$R$  ind. of  $A$ :  $R \perp\!\!\!\perp A$ ,  $P_a(R = 1) = P_b(R = 1) = P(R = 1)$

Approximate versions:

$$\frac{P_a(R=1)}{P_b(R=1)} \geq 1 - \epsilon \quad \text{or} \quad |P_a(R = 1) - P_b(R = 1)| \leq \epsilon$$

+ Legal support ("four-fifth rule")

- Rules out perfect predictor  $R = Y$  when base rates are different.

- Laziness: Criterion can be trivially satisfied if we give loan to qualified people from one group and random people from the other.  $\rightarrow$  May establish a negative track record for 2nd group.

*Example: Acceptance rates of applicants from two groups must be equal, i.e. same percentage of applications receive loans.*

### Second criterion: Separation

$R$  independent of  $A$ , given  $Y$ :  $R \perp\!\!\!\perp A | Y$

**TP:**  $P_a(R = 1|Y = 1) = P_b(R = 1|Y = 1) = P(R = 1|Y = 1)$

**FP:**  $P_a(R = 1|Y = 0) = P_b(R = 1|Y = 0) = P(R = 1|Y = 0)$

**Equality of Opportunity:** Only match TP rate

+ Optimal predictor not ruled out:  $R = Y$  is allowed

+ Penalizes laziness: it provides incentive to reduce errors uniformly in all groups

- May not help closing the gap between the groups

*Example (Equality of Opportunity): Give loan to equal proportion of individuals who would in reality repay.*

### Third criterion: Sufficiency

$Y$  and  $A$  independent given  $R$ :  $Y \perp\!\!\!\perp A | R$ ,  $r \in [0, 1]$

$P_a(Y = 1|R = r) = P_b(Y = 1|R = r) = P(Y = 1|R = r)$

Can be reached by calibration for each group:  $P_a(Y = 1|R = r) = r$

+ Satisfied by Bayes optimal classifier  $r(X, A) = E[Y|X = x, A = a]$

+ For predicting  $Y$  no need to see  $A$  when  $R$  is given

+ Equal chance of success  $Y = 1$  given acceptance  $R = 1$

- May not help closing the gap between the groups

*Example: The score used to determine if a candidate would repay should reflect the candidate's real capability of repay.*

## 13.3 Adversarial Examples / Robustness

**Adversarial Examples:** Deliberate perturbations of the data designed to achieve a specific malicious goal (e.g. cause a misclassification).

**Certifiable robustness** provides mathematical guarantees on defending against Adversarial Examples.

# 1 K-Nearest Neighbor Classification

## Indicator Variable:

$$\mathbb{I}(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases}$$

## Most probable class label $\hat{y}$ :

$$\hat{y} = \underset{c}{\operatorname{argmax}} p(y = c | \mathbf{x}, k)$$

## Classification: $\mathcal{N}_k(\mathbf{x})$ – $k$ nearest neighbors of $\mathbf{x}$

$$p(y = c | \mathbf{x}, k) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \mathbb{I}(y_i = c)$$

## Weighted Classification:

$$p(y = c | \mathbf{x}, k) = \frac{1}{Z} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \frac{1}{d(\mathbf{x}, \mathbf{x}_i)} \mathbb{I}(y_i = c)$$

$$\text{with } Z = \sum_{i \in \mathcal{N}_k(\mathbf{x})} \frac{1}{d(\mathbf{x}, \mathbf{x}_i)}$$


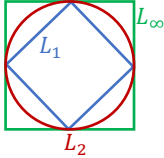



## Regression:

$$\hat{y} = \frac{1}{Z} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \frac{1}{d(\mathbf{x}, \mathbf{x}_i)} y_i, \quad \hat{y} = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x})} y_i$$

## Classification Performance:

Accuracy:	$\text{acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$
Precision:	$\text{prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}$
Sensitivity/Recall/TP-rate:	$\text{rec} = \frac{\text{TP}}{\text{TP} + \text{FN}}$
Specificity/TN-rate:	$\text{tnr} = \frac{\text{TN}}{\text{FP} + \text{TN}}$
False Negative Rate:	$\text{fnr} = \frac{\text{FN}}{\text{TP} + \text{FN}}$
False Positive Rate:	$\text{fpr} = \frac{\text{FP}}{\text{FP} + \text{TN}}$
F1 Score:	$f1 = \frac{2 \cdot \text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}}$

## Distance Measures: $L_\infty \leq L_2 \leq L_1$

$L_1$ norm:		$\sum_i  u_i - v_i $	
$L_2$ norm:		$\sqrt{\sum_i (u_i - v_i)^2}$	
$L_\infty$ norm:		$\max_i  u_i - v_i $	
Mahalanobis:		$\sqrt{(\mathbf{u} - \mathbf{v})^T \Sigma^{-1} (\mathbf{u} - \mathbf{v})}$	
Angle:		$\cos \alpha = \frac{\mathbf{u}^T \mathbf{v}}{\ \mathbf{u}\  \ \mathbf{v}\ }$	

## Data Standardization:

$$\text{Z-Score: } x_{i, \text{std}} = \frac{x_i - \mu_i}{\sigma_i}$$

Many models are sensitive to differences in scale.

## Problem: Curse of dimensionality

Large distance between neighbors ( $N$  has to grow exponentially with the number of features)

## K-Fold Cross-Validation [Decision Trees]:

Split learning data into  $K$  folds (e.g.  $K = 10$ ). Use  $K - 1$  folds for training and 1 fold for evaluation. Average over all folds to get the performance estimate. LOOCV =  $N$ -fold CV  $\rightarrow$  need to train model  $N$  times but interesting if not a lot of training data is available

# 2 Decision Trees

Decision trees (with single feature tests) partition the input space into cuboid regions.

## Inference on decision trees:

1. Test attributes of  $\mathbf{x}$  to find region  $\mathcal{R}$  that contains it and get the class distribution  $\mathbf{n}_{\mathcal{R}} = (n_{c_1, \mathcal{R}}, \dots, n_{c_k, \mathcal{R}})$  for the classes  $\mathcal{C} = \{c_1, \dots, c_k\}$ .

2. Probability of data point  $\mathbf{x} \in \mathcal{R}$  belonging to class  $c$ :

$$p(y = c | \mathcal{R}) = \frac{n_{c, \mathcal{R}}}{\sum_{c_i \in \mathcal{R}} n_{c_i, \mathcal{R}}}$$

3. New unseen sample  $\mathbf{x}$  is given the most common label in its corresponding region  $\mathcal{R}$ .

$$\begin{aligned} \hat{y} &= \underset{c}{\operatorname{argmax}} p(y = c | \mathbf{x}) = \underset{c}{\operatorname{argmax}} p(y = c | \mathcal{R}) \\ &= \underset{c}{\operatorname{argmax}} n_{c, \mathcal{R}} \end{aligned}$$

## Building a decision tree:

Building optimal decision tree is NP-complete  $\rightarrow$  Grow the tree top-down and choose the best split node-by-node using a *greedy heuristic* on training data.

## Improvement heuristic / Information gain:

Split  $s$  of  $t$  into  $t_R$  and  $t_L$  if  $\Delta i(s, t)$  is best improvement:

$$\Delta i(s, t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R)$$

Where  $p_L$  and  $p_R$  are the percentages of original data

## Impurity measures: $\pi_c = p(y = c | t)$

Misclassification rate:  $i_E(t) = 1 - \max_c \pi_c$

Entropy:  $i_H(t) = - \sum_{c_i \in \mathcal{C}} \pi_{c_i} \log_2(\pi_{c_i})$

Gini index:  $i_G(t) = 1 - \sum_{c_i \in \mathcal{C}} \pi_{c_i}^2 = \sum_{c_i \in \mathcal{C}} \pi_{c_i} (1 - \pi_{c_i})$

Entropy and Gini I. prefer very pure splits due to non-linearity.

## Stopping criteria:

- Distribution in branch is pure, i.e.  $i(t) = 0$
- Maximum depth reached
- Number of samples in each branch below  $t_n$
- Benefit of splitting below  $\Delta i(s, t) < t_\Delta$
- Accuracy on validation set

## Reduced error pruning: Grow tree max. and then prune it.

Delete all descendant nodes of  $t$  but not  $t$  itself ( $T \setminus T_t$ : pruned tree):

- Use val. set to get an error estimate:  $\text{err}_{\mathcal{D}_V}(T)$
- For each node  $t$  calculate  $\text{err}_{\mathcal{D}_V}(T \setminus T_t)$
- Prune tree at node with highest error reduction
- Repeat until:  $\text{err}_{\mathcal{D}_V}(T) < \text{err}_{\mathcal{D}_V}(T \setminus T_t)$

## Ensembles:

- **Bagging:** Combining predictions of many classifiers which were created by sampling training set  $\rightarrow$  e.g. Random forests
- **Boosting:** Incrementally train weak classifiers to correct previous mistakes
- **Stacking:** Train a meta-classifier with the base classifier's predictions as features

### 3 Probabilistic Inference

#### General:

Independent and identically distributed (i.i.d.):

$$p(\mathcal{D} | \theta_1, \dots, \theta_N) = \prod_{i=1}^N p(x_i | \theta)$$

Logarithm (monotonic function) preserves critical points:

$$\operatorname{argmax}_{\theta} p(\theta | \mathcal{D}) = \operatorname{argmax}_{\theta} \log p(\theta | \mathcal{D})$$

#### Coin Flip Example:

$$p(T|\theta) = \operatorname{Ber}(f = T|\theta) = \theta, \quad [T \triangleq 1, H \triangleq 0]$$

$$p(\theta|a, b) = \operatorname{Beta}(\theta|a, b)$$

#### Maximum Likelihood Estimation (MLE):

$$\theta_{MLE} = \operatorname{argmax}_{\theta} p(\mathcal{D}|\theta)$$

1. Find matching distribution
2. Apply i.i.d. formula
3. Take logarithm of distribution
4. Derivate w.r.t.  $\theta$
5. Solve for  $\theta$

This is a convex (concave) function of  $\theta$ . To minimize (maximize), compute the derivative, set it to zero and solve for  $\theta$ .

**Problem:** Performs poorly if little data is available (overfitting)

Coin Flip Example:

$$\theta_{MLE} = \frac{|T|}{|T| + |H|}$$

Predict outcome of new flip:

$$p(f_{new} = T | \theta_{MLE}) = \operatorname{Ber}(f_{new} = T | \theta_{MLE}) = \theta_{MLE}$$

#### Maximum A Posteriori Estimation (MAP):

$$\theta_{MAP} = \operatorname{argmax}_{\theta} p(\theta|\mathcal{D})$$

Bayes Formula:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta) \cdot p(\theta)}{p(\mathcal{D})} \propto p(\mathcal{D}|\theta) \cdot p(\theta)$$

1. Find matching distribution
2. Find matching prior
3. Apply i.i.d. formula
4. Take logarithm of distribution
5. Derivate w.r.t.  $\theta$
6. Solve for  $\theta$

Coin Flip Example:

$$\theta_{MAP} = \frac{|T| + a - 1}{|T| + |H| + a + b - 2}$$

Predict outcome of new flip:

$$p(f_{new} = T | \theta_{MAP}) = \operatorname{Ber}(f_{new} = T | \theta_{MAP}) = \theta_{MAP}$$

#### Posterior Distribution Estimate (Fully Bayesian):

Find full distribution  $p(\theta|\mathcal{D})$ , not just point estimate:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta) \cdot p(\theta)}{p(\mathcal{D})}$$

1. Calculate  $p(\mathcal{D}|\theta) \cdot p(\theta)$
2. Find normalization constant  $p(\mathcal{D})$ 
  - a. Computing integral
  - b. Pattern matching (Conjugate Prior?)

Coin Example:  $p(\theta|\mathcal{D}) = \operatorname{Beta}(\theta|a + |T|, b + |H|)$

#### Posterior Predictive Distribution (Fully Bayesian):

$$\begin{aligned} p(f_{new} = T | \mathcal{D}, a, b) &= \int_0^1 p(f_{new} = T, \theta | \mathcal{D}, a, b) d\theta \\ &= \int_0^1 \underbrace{p(f_{new} = T | \theta)}_{\text{Likelihood}} \underbrace{p(\theta | \mathcal{D}, a, b)}_{\text{Posterior}} d\theta \end{aligned}$$

1. Calculate Posterior Distribution Estimate
2. Find matching distribution for likelihood (Coin flip example:  $\operatorname{Ber}(f_{new} = T | \theta)$ )
3. Calculate integral

Coin Flip Example - Predict outcome of new flip:

$$p(f_{new} = T | \mathcal{D}, a, b) = \operatorname{Ber}\left(f_{new} = T \mid \frac{|T|+a}{|T|+|H|+a+b}\right)$$

#### Overview: Coin Flip vs. Regression

	train data	likelihood	prior	posterior
coin:	$\mathcal{D} = \mathbf{X}$	$p(\mathcal{D}   \theta)$	$p(\theta   a, b)$	$p(\theta   \mathcal{D})$
regr.:	$\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$	$p(\mathbf{y}   \mathbf{X}, \mathbf{w}, \beta)$	$p(\mathbf{w}   \cdot)$	$p(\mathbf{w}   \mathbf{X}, \mathbf{y}, \beta, \cdot)$

#### Conjugate Prior:

If a prior is conjugate for the given likelihood, then the posterior will be of the same family as the prior. E.g. Beta distribution is a conjugate prior for the Bernoulli likelihood.

**Posterior**  $\propto$  Likelihood  $\cdot$  **Prior**

**Beta**  $\propto$  Bernoulli/Binomial  $\cdot$  **Beta**

**Gaussian**  $\propto$  Gaussian  $\cdot$  **Gaussian**

#### Hoeffding's Inequality:

How many flips  $N$  are needed to know with probability  $1 - \delta$  that the error  $|\theta_{MLE} - \theta|$  is smaller than  $\epsilon$ ?

$$N \geq \frac{\ln(2/\delta)}{2\epsilon^2}$$



## 4 Linear Regression

**Linear dependencies:**  $\mathbf{x}, \mathbf{w} \in \mathbb{R}^D$ ,  $\hat{\mathbf{y}} \in \mathbb{R}^N$  (bias absorbed)

$$\hat{\mathbf{y}} = f_{\mathbf{w}}(\mathbf{x}) = \sum_{j=1}^D w_j x_j = \mathbf{w}^T \mathbf{x}, \quad \hat{\mathbf{y}} = \mathbf{X} \mathbf{w}$$

Observation matrix:  $X_{ij} = x_{ij}$ ,  $\mathbf{X} \in \mathbb{R}^{N \times D}$

**Non-linear dependencies:**  $\phi(\mathbf{x}), \mathbf{w} \in \mathbb{R}^M$

$$\hat{\mathbf{y}} = f_{\mathbf{w}}(\mathbf{x}) = \sum_{j=1}^M w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}), \quad \hat{\mathbf{y}} = \Phi \mathbf{w}$$

Design matrix:  $\Phi_{ij} = \phi_j(\mathbf{x}_i)$ ,  $\Phi \in \mathbb{R}^{N \times M}$  (basis functions)

If  $\phi(\mathbf{x}) = \mathbf{A}^T \mathbf{x}$ , then  $\Phi = \mathbf{X} \cdot \mathbf{A}$ .

**Bias-variance trade-off:**

- High bias (expected error due to model mismatch)  $\rightarrow$  model too rigid (underfitting)
- High variance (variation due to randomness in training data)  $\rightarrow$  model too flexible  $\rightarrow$  captures noise (overfitting)

### 4.1 Ordinary Least Squares Regression

**Loss Function:**

$$\begin{aligned} E_{LS}(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^N (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 \\ &= \frac{1}{2} (\Phi \mathbf{w} - \mathbf{y})^T (\Phi \mathbf{w} - \mathbf{y}) \\ &= \frac{1}{2} (\mathbf{X} \mathbf{w} + w_0 \mathbf{1}_N - \mathbf{y})^T (\mathbf{X} \mathbf{w} + w_0 \mathbf{1}_N - \mathbf{y}) \end{aligned}$$

**Optimal Weight Vector / Normal Equation:**

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} E_{LS}(\mathbf{w}) = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y} = \Phi^\dagger \mathbf{y}$$

**Probabilistic Interpretation:**

Least squares regression is equivalent to maximum likelihood estimation and to sampling from i.i.d. samples with Gaussian error term.

$$y_i = f_{\mathbf{w}}(\mathbf{x}_i) + \epsilon_i \sim \mathcal{N}(f_{\mathbf{w}}(\mathbf{x}_i), \beta^{-1}), \quad \epsilon_i: \text{noise}$$

$$p(y_i | f_{\mathbf{w}}(\mathbf{x}_i), \beta) = \mathcal{N}(y_i | f_{\mathbf{w}}(\mathbf{x}_i), \beta^{-1})$$

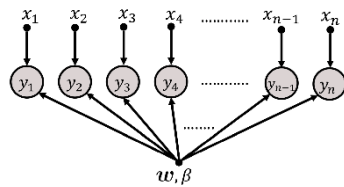
$$p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^N p(y_i | f_{\mathbf{w}}(\mathbf{x}_i), \beta) = \mathcal{N}(\mathbf{y} | \Phi \mathbf{w}, \beta^{-1} \mathbf{I})$$

$$\begin{aligned} E_{ML}(\mathbf{w}, \beta) &= -\ln p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \beta) \\ &= \frac{\beta}{2} \sum_{i=1}^N (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 - \frac{N}{2} \ln(\beta) + \frac{N}{2} \ln(2\pi) \end{aligned}$$

$$\begin{aligned} \mathbf{w}_{ML} &= \underset{\mathbf{w}}{\operatorname{argmax}} p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \beta) = \underset{\mathbf{w}}{\operatorname{argmin}} E_{ML}(\mathbf{w}) \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} E_{LS}(\mathbf{w}) \end{aligned}$$

$$\beta_{ML} = \underset{\beta}{\operatorname{argmin}} E_{ML}(\mathbf{w}_{ML}, \beta)$$

$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{i=1}^N (\mathbf{w}_{ML}^T \phi(\mathbf{x}_i) - y_i)^2$$



### 4.2 Ridge Regression

**Loss Function:**

$$\begin{aligned} E_{Ridge}(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \\ &= \frac{1}{2} (\Phi \mathbf{w} - \mathbf{y})^T (\Phi \mathbf{w} - \mathbf{y}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \end{aligned}$$

**Optimal Weight Vector / Normal Equation:**

$$\begin{aligned} \mathbf{w}^* &= \underset{\mathbf{w}}{\operatorname{argmin}} E_{Ridge}(\mathbf{w}) \\ &= (\Phi^T \Phi + \lambda \mathbf{I}_M)^{-1} \Phi^T \mathbf{y} \end{aligned}$$

$$\begin{aligned} \nabla_{\mathbf{w}} E_{LS}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[ \frac{1}{2} (\Phi \mathbf{w} - \mathbf{y})^T (\Phi \mathbf{w} - \mathbf{y}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \right] \\ &= \nabla_{\mathbf{w}} \left[ \frac{1}{2} (\mathbf{w}^T \Phi^T \Phi \mathbf{w} - 2 \mathbf{w}^T \Phi^T \mathbf{y} + \mathbf{y}^T \mathbf{y}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \right] \\ &= \Phi^T \Phi \mathbf{w} - \Phi^T \mathbf{y} + \lambda \mathbf{w} \\ &= (\Phi^T \Phi + \lambda \mathbf{I}_M) \mathbf{w} - \Phi^T \mathbf{y} \end{aligned}$$

**Probabilistic Interpretation:**

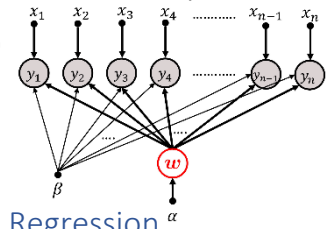
Ridge regression is equivalent to Maximum a posteriori estimation with Gaussian prior  $p(\mathbf{w})$ . [ $\beta, \alpha$  given]

$$p(\mathbf{w} | \alpha) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I}) = \left( \frac{\alpha}{2\pi} \right)^{\frac{M}{2}} \exp \left( -\frac{\alpha}{2} \mathbf{w}^T \mathbf{w} \right)$$

$$p(\mathbf{w} | \mathbf{X}, \mathbf{y}, \beta, \alpha) = \frac{p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \beta) \cdot p(\mathbf{w} | \alpha)}{p(\mathbf{y} | \mathbf{X}, \beta, \alpha)}$$

$$\begin{aligned} E_{MAP}(\mathbf{w}) &= -\ln p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \beta) - \ln p(\mathbf{w} | \alpha) \\ &= E_{ML}(\mathbf{w}, \beta) - \frac{M}{2} \ln \left( \frac{\alpha}{2\pi} \right) + \frac{\alpha}{2} \|\mathbf{w}\|_2^2 \\ &\propto E_{Ridge}(\mathbf{w}, \beta) + \text{const}, \quad \text{with } \lambda = \frac{\alpha}{\beta} \end{aligned}$$

$$\begin{aligned} \mathbf{w}_{MAP} &= \underset{\mathbf{w}}{\operatorname{argmax}} p(\mathbf{w} | \mathbf{X}, \mathbf{y}, \beta, \alpha) \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} E_{MAP}(\mathbf{w}) \end{aligned}$$



### 4.3 Weighted Least Squares Regression

**Loss Function:**

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^N t_i (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 \\ &= \frac{1}{2} (\Phi \mathbf{w} - \mathbf{y})^T \mathbf{T} (\Phi \mathbf{w} - \mathbf{y}), \quad \mathbf{T} = \operatorname{diag}(t_1, \dots, t_N) \end{aligned}$$

**Optimal Weight Vector / Normal Equation:**

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} E(\mathbf{w}) = (\Phi^T \mathbf{T} \Phi)^{-1} \Phi^T \mathbf{T} \mathbf{y}$$

**Probabilistic Interpretation:**

Weighted least squares is equivalent to probabilistic least squares where we choose  $\beta = t_i$ , therefore making the regression targets no longer identically distributed but still independent.  $y_i \sim \mathcal{N}(\mathbf{w}^T \phi(\mathbf{x}_i), t_i^{-1})$

### 4.4 Fully Bayesian Posterior Distribution

$$p(\mathbf{w} | \mathcal{D}) = p(\mathbf{w} | \mathbf{X}, \mathbf{y}, \beta, \alpha) \propto p(\mathbf{y} | \mathbf{X}, \mathbf{w}, \beta) \cdot p(\mathbf{w} | \alpha)$$

If  $p(\mathbf{w} | \alpha) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I})$  with  $\lambda = \frac{\alpha}{\beta}$ :

$$p(\mathbf{w} | \mathcal{D}) = \mathcal{N}(\mathbf{w} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\text{with } \boldsymbol{\mu} = \mathbf{w}_{MAP} = \beta \boldsymbol{\Sigma} \Phi^T \mathbf{y} \text{ and } \boldsymbol{\Sigma}^{-1} = \alpha \mathbf{I} + \beta \Phi^T \Phi$$

### 4.5 Predicting new Data

Maximum likelihood:  $\mathbf{w}_{ML}$  and  $\beta_{ML}$

$$p(\hat{y}_{new} | \mathbf{x}_{new}, \mathbf{w}_{ML}, \beta_{ML}) = \mathcal{N}(\hat{y}_{new} | \mathbf{w}_{ML}^T \phi(\mathbf{x}_{new}), \beta_{ML}^{-1})$$

Maximum a posteriori:  $\mathbf{w}_{MAP}$

$$p(\hat{y}_{new} | \mathbf{x}_{new}, \mathbf{w}_{MAP}, \beta) = \mathcal{N}(\hat{y}_{new} | \mathbf{w}_{MAP}^T \phi(\mathbf{x}_{new}), \beta^{-1})$$

Posterior predictive distribution:

$$\begin{aligned} p(\hat{y}_{new} | \mathbf{x}_{new}, \mathcal{D}) &= \int p(\hat{y}_{new}, \mathbf{w} | \mathbf{x}_{new}, \mathcal{D}) d\mathbf{w} \\ &= \int p(\hat{y}_{new} | \mathbf{x}_{new}, \mathbf{w}) p(\mathbf{w} | \mathcal{D}) d\mathbf{w} \\ &= \mathcal{N}(\hat{y}_{new} | \boldsymbol{\mu}^T \phi(\mathbf{x}_{new}), \beta^{-1} + \phi(\mathbf{x}_{new})^T \boldsymbol{\Sigma} \phi(\mathbf{x}_{new})) \\ &\rightarrow \text{Variance depends on } \mathbf{x}_{new} \end{aligned}$$

## 5 Linear Classification

### 5.1 Perceptron / Hard decision based classifier

**Decision rule:**  $\mathbf{w}, \mathbf{x} \in \mathbb{R}^D$

$$\hat{y} = f(\mathbf{w}^T \mathbf{x} + w_0) \quad \text{with} \quad f(t) = \begin{cases} 1 & \text{if } t > 0 \\ -1 & \text{otherwise} \end{cases}$$

**Zero-one loss:** Number of misclassified samples

$$l_{01}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^N \mathbb{I}(\hat{y}_i \neq y_i) \quad \mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^N$$

**Linear separability:**

A data set  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$  is linearly separable if there exists a hyperplane for which all  $\mathbf{x}_i$  with  $y_i = 0$  are on one and all  $\mathbf{x}_i$  with  $y_i = 1$  on the other side. Non-linear transformations (basis functions) can map samples into linear separable space.

**Learning task:** SGD with mini-batch size 1

$$\arg\min_{\mathbf{w}, b} \sum_i L(y_i, \mathbf{w}^T \mathbf{x}_i + w_0) = \sum_i \max(0, \epsilon - y_i(\mathbf{w}^T \mathbf{x}_i + w_0))$$

**Hinge loss:**

$$L(u, v) = \max(0, \epsilon - uv) = \begin{cases} \epsilon - uv, & \text{if } uv < \epsilon \\ 0 & , \quad \text{else} \end{cases}$$

**Learning rule:** For misclassified samples

$$\mathbf{w} \leftarrow \begin{cases} \mathbf{w} + \tau n \cdot \mathbf{x}_i & \text{if } y_i = 1 \\ \mathbf{w} - \tau n \cdot \mathbf{x}_i & \text{if } y_i = -1 \end{cases} \quad \& \quad w_0 \leftarrow \begin{cases} w_0 + \tau n & \text{if } y_i = 1 \\ w_0 - \tau n & \text{if } y_i = -1 \end{cases}$$

e.g.  $\tau n = 1$ ,  $\tau$ : learning rate,  $n$ : # samples

→ Continue until all samples are classified correctly.

**Multiple classes:**

- One-Versus-Rest: Train binary classifiers class  $C_i \leftrightarrow$  not class  $C_i \rightarrow$  each hyperplane  $\mathcal{H}_i$  makes a decision
- One-Versus-One: Train binary classifiers class  $C_i \leftrightarrow$  not class  $C_i$  and use majority vote to classify.
- Multiclass Discriminant: Define  $\mathcal{C}$  functions  $f_c(\mathbf{x}) = \mathbf{w}_c^T \mathbf{x} + w_{0c}$  with the decision rule  $\hat{y} = \arg\max_{c \in \mathcal{C}} f_c(\mathbf{x})$ .

**Limitations of hard-decision based classifiers:**

No measure of uncertainty. Cannot handle noisy data.

Poor generalization. Difficult to optimize.

### 5.2 Probabilistic Generative Model

Model the joint distribution:

$$\underbrace{p(\mathbf{y} = c, \mathbf{x} | \psi, \theta)}_{\text{class posterior}} = \underbrace{p(\mathbf{x} | \mathbf{y} = c, \psi)}_{\text{class conditional}} \cdot \underbrace{p(\mathbf{y} = c | \theta)}_{\text{class prior}}$$

**Class Prior:**

Categorical distribution:

$$p(\mathbf{y}) = \prod_{c=1}^C \theta_c^{\mathbb{I}(\mathbf{y}=c)} \quad \text{or} \quad p(\mathbf{y} = c) = \theta_c, \quad \boldsymbol{\theta} \in \mathbb{R}^C$$

**MLE of  $\boldsymbol{\theta}$ :**  $\boldsymbol{\pi} \in \mathbb{R}^C$

$$\theta_c^{MLE} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(y_i = c) = \frac{N_c}{N} = \pi_c$$

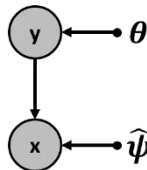
**Class Conditional:**  $\mathbf{x}, \boldsymbol{\mu}_c \in \mathbb{R}^D, \boldsymbol{\Sigma}_c \in \mathbb{R}^{D \times D}$

Multivariate normal with global covariance:

$$p(\mathbf{x} | \mathbf{y} = c) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_c|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)\right)$$

**MLE of  $\boldsymbol{\mu}_c$ :**

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{\mathbf{y}^{(n)=c}}^N \mathbf{x}^{(n)}$$



**MLE:**

$$\boldsymbol{\mu}_c^*, \boldsymbol{\Sigma}_c^*, \theta_c^* = \arg\max_{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, \theta_c} \log p(\mathbf{X}, \mathbf{y} | \{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, \theta_c\}_{c=1}^C)$$

$$p(\mathbf{X}, \mathbf{y} | \{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, \theta_c\}_{c=1}^C) = \prod_{i=1}^N p(\mathbf{x}_i, y_i | \{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, \theta_c\}_{c=1}^C) = \prod_{i=1}^N \prod_{c=1}^C [p(\mathbf{x}_i, y = c | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, \theta_c)]^{y_{ic}}$$

Constraint:  $\prod_{c=1}^C \theta_c = 1$

**Inference:**

$$p(y = c | \mathbf{x}, \psi, \theta) \propto p(y = c, \mathbf{x} | \psi, \theta)$$

$$\hat{y}_{new} = \arg\max_{c \in \mathcal{C}} (y = c | \mathbf{x}_{new}, \psi, \theta)$$

**Two classes (binary):**  $a = 0$ : decision boundary

$$p(y | \mathbf{x}) = \text{Ber}(y | p(y = 1 | \mathbf{x}))$$

$$p(y = 1 | \mathbf{x}) = \sigma(a) = \frac{1}{1 + \exp(-a)} = \frac{p(\mathbf{x} | y = 1)p(y = 1)}{p(\mathbf{x} | y = 1)p(y = 1) + p(\mathbf{x} | y = 0)p(y = 0)}$$

$$\text{where: } a = \log \frac{p(\mathbf{x} | y = 1)p(y = 1)}{p(\mathbf{x} | y = 0)p(y = 0)}$$

**More classes:**

$$p(y = c | \mathbf{x}) = \text{Cat}(y = c | p(\mathbf{y} | \mathbf{x}))$$

$$p(y = c | \mathbf{x}) = \frac{p(\mathbf{x} | y = c)p(y = c)}{\sum_{c'=1}^C p(\mathbf{x} | y = c')p(y = c')}$$

#### 5.2.1 Linear Discriminant Analysis (LDA)

Gaussian class conditionals with shared covariance matrix ( $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_c, \forall c \in \mathcal{C}$ ).

**MLE for shared covariance matrix  $\boldsymbol{\Sigma}$ :**

$$\boldsymbol{\Sigma} = \sum_{c=1}^C \frac{N_c}{N} \mathbf{S}_c \quad \text{with} \quad \mathbf{S}_c = \frac{1}{N_c} \sum_{\substack{n=1 \\ \mathbf{y}^{(n)=c}}^N (\mathbf{x}^{(n)} - \boldsymbol{\mu}_c)(\mathbf{x}^{(n)} - \boldsymbol{\mu}_c)^T$$

**LDA for two classes:** Linear Decision Boundary

$$a = \mathbf{w}^T \mathbf{x} + w_0, \quad \mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

$$w_0 = -\frac{1}{2} \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0 + \log \frac{p(y = 1)}{p(y = 0)}$$

**LDA for more classes:** Linear Decision Boundaries

$$p(y = c | \mathbf{x}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x} + w_{c0})}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x} + w_{c'0})} = \sigma_c(\mathbf{W} \mathbf{x} + \mathbf{w}_0)$$

$$\mathbf{w}_c = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c \quad w_{c0} = -\frac{1}{2} \boldsymbol{\mu}_c^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c + \log p(y = c)$$

Decision Boundaries:  $\mathbf{w}_1^T \mathbf{x} + w_{10} = \mathbf{w}_2^T \mathbf{x} + w_{20}$

### 5.2.2 Naive Bayes Classification

Diagonal per-class covariance matrices  $\Sigma_c$  (if all  $p(x_i|y=c) \sim \mathcal{N}$ ). Advantage: Naïve Bayes can handle mixed data types.

Assumption:  $p(x_1, x_2, \dots, x_d|y=c) = \prod_{i=1}^d p(x_i|y=c)$   
(Class conditional: The features of the samples are conditionally independent given the class.)

**NBC for two classes:** Quadratic Decision Boundaries

$$a = \mathbf{x}^T \mathbf{W}_2 \mathbf{x} + \mathbf{w}_1^T \mathbf{x} + w_0$$

$$\mathbf{W}_2 = \frac{1}{2} [\Sigma_0^{-1} - \Sigma_1^{-1}] \quad \mathbf{w}_1 = \Sigma_1^{-1} \mu_1 - \Sigma_0^{-1} \mu_0$$

$$w_0 = -\frac{1}{2} \mu_1^T \Sigma_1^{-1} \mu_1 + \frac{1}{2} \mu_0^T \Sigma_0^{-1} \mu_0 + \log \frac{\pi_1}{\pi_0} + \frac{1}{2} \log \frac{|\Sigma_0|}{|\Sigma_1|}$$

### 5.3 Ordinary 2-class Logistic Regression

Directly model the posterior distribution  $p(y|\mathbf{x}) = \text{Ber}(y|\sigma(\mathbf{w}^T \mathbf{x}))$  by treating  $\mathbf{w}$  and  $w_0$  as free parameters ( $w_0$  absorbed).

$$p(y=1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) \quad p(y=0|\mathbf{x}) = 1 - \sigma(\mathbf{w}^T \mathbf{x})$$

**Likelihood:** i.i.d. assumption,  $y_i \in \{0,1\}$ :

$$\begin{aligned} p(\mathbf{y}|\mathbf{w}, \mathbf{X}) &= \prod_{i=1}^N p(y_i|\mathbf{x}_i, \mathbf{w}) = \\ &= \prod_{i=1}^N p(y_i=1|\mathbf{x}_i, \mathbf{w})^{y_i} (1 - p(y_i=1|\mathbf{x}_i, \mathbf{w}))^{1-y_i} = \\ &= \prod_{i=1}^N \sigma(\mathbf{w}^T \mathbf{x}_i)^{y_i} (1 - \sigma(\mathbf{w}^T \mathbf{x}_i))^{1-y_i} \end{aligned}$$

With labels  $y_i \in \{-1,1\}$ :

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}) = \prod_{i=1}^N \sigma(y_i(\mathbf{w}^T \mathbf{x}_i))$$

**Loss Function: Binary Cross Entropy**

Negative log-likelihood of Binary Logistic Regression

$$\begin{aligned} E(\mathbf{w}) &= -\log p(\mathbf{y}|\mathbf{w}, \mathbf{X}) \\ &= -\sum_{i=1}^N (y_i \log p(y_i=1|\mathbf{x}_i, \mathbf{w}) + (1-y_i) \log(1 - p(y_i=1|\mathbf{x}_i, \mathbf{w}))) \\ &= -\sum_{i=1}^N (y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i) + (1-y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))) \end{aligned}$$

**Gradient of Loss Function:**

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \nabla_{\mathbf{w}} (-\ln p(\mathbf{y}|\mathbf{w}, \mathbf{X})) = \sum_{i=1}^N \mathbf{x}_i (\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i)$$

### 5.4 2-class Logistic Ridge Regression

**Loss Function: Binary Cross Entropy + weights regular.**

$$E_{\text{Ridge}}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_q^q$$

For  $q=2$ : Corresponds to MAP estimation with a Gaussian prior on  $\mathbf{w}$  with precision  $\lambda$ .  $E(\mathbf{w})$  (with and without regularization) is a convex function.

**Gradient of Loss Function:**

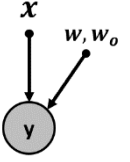
$$\nabla_{\mathbf{w}} E_{\text{Ridge}}(\mathbf{w}) = \sum_{i=1}^N \mathbf{x}_i (\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i) + \lambda \mathbf{w}$$

### 5.5 Ordinary Multi-Class Logistic Regression

**Likelihood: Softmax function**

$$p(y=c|\mathbf{x}) = \sigma_c(\mathbf{W}\mathbf{x}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'} \exp(\mathbf{w}_{c'}^T \mathbf{x})}$$

where  $\mathbf{W} \in \mathbb{R}^{C \times D}$ ,  $\mathbf{x}, \mathbf{w}_c \in \mathbb{R}^D$



**Loss Function: Cross Entropy**

Negative log-likelihood of Multiclass Logistic Regression

$$p(\mathbf{Y}|\mathbf{w}, \mathbf{X}) = -\prod_{i=1}^N \prod_{c=1}^C (p(y_i=c|\mathbf{x}_i, \mathbf{w}))^{y_{ic}}$$

$$E(\mathbf{w}) = -\log(p(\mathbf{Y}|\mathbf{w}, \mathbf{X})) = -\sum_{i=1}^N \sum_{c=1}^C y_{ic} \log p(y_i=c|\mathbf{x}_i, \mathbf{w})$$

One-Hot Encoding:  $\mathbf{Y} \in \{0,1\}^{N \times C}$  with

$$y_{ic} = \begin{cases} 1, & \text{if sample } i \text{ belongs to class } c \\ 0, & \text{else} \end{cases}$$

### 5.6 Optimizing Logistic Regression

**Gradient Descent:**

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \tau \cdot \nabla E(\mathbf{w}_t)$$

### 5.7 Sigmoid and Softmax

**Softmax:**

$$\sigma_i(\mathbf{x}) = \frac{\exp x_i}{\sum_{k=1}^K \exp x_k}, \quad \mathbf{x} \in \mathbb{R}^K$$

$$\frac{\partial \sigma_i(\mathbf{x})}{\partial x_j} = \sigma_i(\mathbf{x}) \frac{\partial \log \sigma_i(\mathbf{x})}{\partial x_j} = \sigma_i(\mathbf{x}) (\delta_{ij} - \sigma_j(\mathbf{x}))$$

**Sigmoid:**

$$\sigma(a) = \frac{1}{1 + \exp(-a)} = \frac{\exp(a)}{1 + \exp(a)}$$

$$\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$$

$$\sigma(-a) = 1 - \sigma(a) = \frac{1}{1 + \exp(a)} \quad a = \ln \frac{\sigma}{1 - \sigma}$$

$$\tanh(a) = 2\sigma(2a) - 1 = \frac{\exp(2a) - 1}{\exp(2a) + 1}$$

### 5.8 Generative vs. Discriminative Models

- Discriminative models achieve better performance in pure classification tasks because no assumptions about the class distributions have to be made.
- Generative models are fragile when assumptions are violated.
- Generative models handle missing data and outliers better, can generate new data and are more appropriate in the semi-supervised setting.



## 6 Optimization

### 6.1 Convexity

**Convex sets:**  $X$  is convex iff for all  $x, y \in X$ :

$$\lambda x + (1 - \lambda)y \in X \text{ for } \lambda \in [0, 1]$$

$X$  is not convex iff we can choose two points in  $X$  where the line connecting the points does not completely reside in  $X$ .  
The intersection of convex sets (e.g. half-spaces) is convex.

**Extreme points / vertices of a convex set  $X$ :**

$x \in X$  is a vertex of  $X$  if for arbitrary  $\lambda > 1$  and arbitrary  $y \in X$  (with  $y \neq x$ ) it holds that  $\lambda x + (1 - \lambda)y \notin X$ .

**Convex functions:**

$f$  is convex on the convex set  $X$  iff for all  $x, y \in X$ :

$$\lambda f(x) + (1 - \lambda)f(y) \geq f(\lambda x + (1 - \lambda)y) \text{ for } \lambda \in [0, 1]$$

Each local minimum is a global minimum.

**First order convexity conditions:**

$f$  is convex on the convex set  $X$  iff for all  $x, y \in X$ :

$$f(y) - f(x) \geq \frac{f((1 - t)x + ty) - f(x)}{t}$$
$$f(y) \geq (y - x)^T \nabla f(x) + f(x)$$

**Second order convexity conditions:**

$f$  is convex on  $X$  iff for all  $x \in X$ :

$$\nabla_x^2 f(x) \text{ is PSD} \quad \text{or} \quad \frac{\partial^2 f(x)}{\partial x^2} \geq 0$$

**Convexity preserving operations:**

Let  $f_1, f_2$  are convex and  $g$  is concave, then:

- 1)  $h(x) = f_1(x) + f_2(x)$  is convex
- 2)  $h(x) = \max\{f_1(x), f_2(x)\}$  is convex
- 3)  $h(x) = c \cdot f_1(x)$  is convex if  $c \geq 0$
- 4)  $h(x) = c \cdot g(x)$  is convex if  $c \leq 0$
- 5)  $h(x) = f_1(Ax + b)$  is convex ( $A$  matrix,  $b$  vector)
- 6)  $h(x) = m(f_1(x))$  is convex if  $m: \mathbb{R} \rightarrow \mathbb{R}$  is convex and nondecreasing

$$\max\{f_1(x), f_2(x)\} = -\min\{-f_1(x), -f_2(x)\}$$

### 6.2 Gradient Descent

- Only for differentiable functions
- Gradient points into steepest ascent direction
- Locally, the gradient is a good approximation of the objective function

**GD with Line Search:**

**given** a starting point  $\theta \in \text{Dom}(f)$

**repeat**

1.  $\Delta\theta := -\nabla f(\theta)$
2. Line search:  $t^* = \underset{t > 0}{\operatorname{argmin}} f(\theta + t \cdot \Delta\theta)$
3. Update.  $\theta := \theta + t^* \cdot \Delta\theta$

**until** stopping criterion is satisfied.

**Learning Rate to avoid Line Search:**

$$\theta_{t+1} \leftarrow \theta_t - \tau \cdot \nabla f(\theta_t)$$

- $\tau$  too small: slowly converging or ending up in saddle point or local minima
- $\tau$  too high: algorithm might oscillate, no convergence  
→ Learning Rate Schedule: Learning rate as a decreasing function  $\tau_t$  of the iteration number  $t$ . Convergence easily guaranteed if  $\lim_{t \rightarrow \infty} \tau_t = 0$ .

**Learning rate adaption:**

**Momentum:**

Integrating gradient history into parameters → Search accelerates as long as gradient points to same direction.

**AdaGrad:**

Different learning rates per parameter. Learning rate depends inversely on accumulated “strength” of all previously computed gradients.

**Adam:**

Combination of several LR adaption “tricks” (e.g. first and second momentum).

**Stochastic Gradient Descent (SGD):**

Use a mini-batch of entire data to compute a noisy gradient and use it for parameter updating.

1. Randomly pick a (small) subset  $S$  from the entire data  $\mathcal{D}$ , the so called mini-batch
2. Compute gradient based on mini-batch
3. Update:  $\theta_{t+1} \leftarrow \theta_t - \tau \cdot \frac{n}{|S|} \cdot \nabla f(\theta_t)$   
 $\nabla f(\theta_t) = \sum_{j \in S} L_j(\theta_t)$ ,  $n$ : total number of samples
4. Pick a new subset and repeat with 2

### 6.3 Newton-Raphson Method

Higher-order optimization technique. Replace learning rate by second derivative. Better performance but more costly computation → only for low dimensional problems.  
Goal:  $\nabla f(\theta) = 0$

**Taylor-Expansion of  $f$  at point  $\theta_t$ :**

$$f(\theta_t + \delta) = f(\theta_t) + \delta^T \nabla f(\theta_t) + \frac{1}{2} \delta^T \nabla^2 f(\theta_t) \delta + O(\delta^3)$$

**Minimize Approximation:**  $\nabla_\delta f(\theta_t + \delta) \stackrel{!}{=} 0$

$$\theta_{t+1} \leftarrow \theta_t - [\nabla^2 f(\theta_t)]^{-1} \nabla f(\theta_t) = \theta_t - \frac{f'(\theta_t)}{f''(\theta_t)}$$

### 6.4 Distributed Learning

→ Exploit multiple machines

**Data Parallelism:**

Use multiple model replicas to preprocess different examples at the same time

**Model Parallelism:**

Many models have lots of inherent parallelism as for example *local connectivity* (as found in CNNs).

## 7 Deep Learning 1

### 7.1 Fully Connected Networks

**MLP with two Hidden Layers:**  $\sigma_i$ : arbitrary activation functions

$$f(\mathbf{x}, \mathbf{W}) = \sigma_2 \left( \mathbf{W}_2 \sigma_1 \left( \mathbf{W}_1 \sigma_0 (\mathbf{W}_0 \mathbf{x}) \right) \right), \mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\}$$

$$\mathbf{X} \in \mathbb{R}^{N \times D}: f(\mathbf{X}, \mathbf{W}) = \sigma_2 \left( \sigma_1 \left( \sigma_0 (\mathbf{X} \mathbf{W}_0^T) \mathbf{W}_1^T \right) \mathbf{W}_2^T \right)$$

### 7.2 Loss Functions

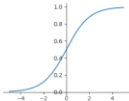
Non-linear activation functions for classification tasks and no activation function for regression tasks.

Prediction target	Output distribution	Final layer	Loss function
Binary	Bernoulli	Sigmoid	Binary cross entropy
Discrete	Categorical	Softmax	Cross entropy
Continuous	Gaussian	Linear	Squared error
Continuous	Arbitrary	GAN, VAE, ...	Various

### 7.3 Activation Functions

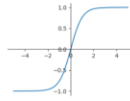
**Sigmoid**

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



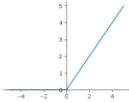
**tanh**

$$\tanh(x)$$



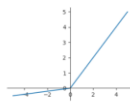
**ReLU**

$$\max(0, x)$$



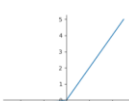
**Leaky ReLU**

$$\max(0.1x, x)$$



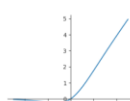
**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



**Swish**

$$x \cdot \sigma(x)$$



→ Gradients may vanish when using saturating activations (e.g. sigmoid)

### 7.4 Parameter Learning with Backpropagation

**Gradient Descent:**

$$\mathbf{W}^{(new)} = \mathbf{W}^{(old)} - \tau \cdot \nabla_{\mathbf{W}} E(\mathbf{W}^{(old)})$$

**Backpropagation:**

**Forward pass:** Evaluate function values of each module and cache the intermediate values.

**Backward pass:** Compute gradients w.r.t. parameters by using chain rule.

**Advantages:** Reuse computations of common ancestors; Only pass through the computation graph twice; Modular structure

**Modules:** Each module in the comp. graph defines:

Forward( $\mathbf{x}$ ): given input  $\mathbf{x}$ , compute output  $\mathbf{y}$

Backward( $\frac{\partial E}{\partial \mathbf{y}}$ ): given the incoming global derivative  $\frac{\partial E}{\partial \mathbf{y}}$

$$\text{compute the product } \frac{\partial E}{\partial \mathbf{x}} = \frac{\partial E}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

### 7.5 Affine Layer

$$\mathbf{a}_n = \mathbf{W} \mathbf{x}_n + \mathbf{b} \text{ where } \mathbf{W} \in \mathbb{R}^{H \times D}, \mathbf{x}_n \in \mathbb{R}^D, \mathbf{a}, \mathbf{b} \in \mathbb{R}^H$$

$$\frac{\partial E}{\partial \mathbf{W}} = \mathbf{x}_n \frac{\partial E}{\partial \mathbf{a}}, \quad \frac{\partial E}{\partial \mathbf{x}_n} = \frac{\partial E}{\partial \mathbf{a}} \mathbf{W}, \quad \frac{\partial E}{\partial \mathbf{b}} = \frac{\partial E}{\partial \mathbf{a}}$$

$$\mathbf{A} = \mathbf{X} \mathbf{W} + \mathbf{1}_N \mathbf{b}^T$$

where  $\mathbf{A} \in \mathbb{R}^{N \times H}$ ,  $\mathbf{X} \in \mathbb{R}^{N \times D}$ ,  $\mathbf{W} \in \mathbb{R}^{D \times H}$ ,  $\mathbf{b} \in \mathbb{R}^H$

$$\frac{\partial E}{\partial \mathbf{W}} = \frac{\partial E}{\partial \mathbf{A}} \mathbf{X}, \quad \frac{\partial E}{\partial \mathbf{X}} = \mathbf{W} \frac{\partial E}{\partial \mathbf{A}}, \quad \frac{\partial E}{\partial \mathbf{b}} = \mathbf{1}_N^T \frac{\partial E}{\partial \mathbf{A}}$$

### 7.6 Numerical stability

$$\sigma_i(\mathbf{x}) = \frac{\exp(x_i)}{\sum_{c=1}^C \exp(x_c)} = \frac{\exp(x_i - a)}{\sum_{c=1}^C \exp(x_c - a)}, \text{ often: } a = \max_i(x_i)$$

$$\log(\sum e^{x_i}) = a + \log(\sum e^{x_i - a})$$

## 8 Deep Learning 2

### 8.1 Convolution Neural Network (CNN)

**Cross Correlation / 2D-Convolution:**  $L \times M$  filter

$$\hat{x}(i, j) = \sum_{l=1}^L \sum_{m=1}^M x(i + l, j + m) k(l, m)$$

**Padding:** Reduce the output (*Valid*) or pad the input (e.g. with zeros) to preserve input size (*Same*, add  $P = \lfloor K/2 \rfloor$  values per side) or to increase input size (*Full*, add  $K - 1$  values).  $K$ : Kernel width,  $D_i$ : input size along a dimension

**Stride  $S$ :** Distance between positions the kernel is applied.

$$\text{Stride } S > 1 \text{ is downsampling. } D_{l+1} = \left\lfloor \frac{D_l + 2P - K}{S} \right\rfloor + 1$$

**Pooling:** Calculate summary statistics in sliding window (e.g. max / mean pooling,  $L_p$  norm pooling)

### 8.2 Training Deep Neural Networks

**Weight symmetry:** Two hidden units have the same bias and weights → same gradient → extract same features → break symmetry by initializing small random values

**Weight scale:** big fan-in + small changes in many weights or big fan-out + backward pass → overshoot → vanishing or exploding gradients

**Xavier Glorot initialization:** Preserve the signal's mean and variance by using weight matrices with zero mean and variance  $\text{Var}(\mathbf{W}) = \frac{2}{\text{fan-in} + \text{fan-out}}$ . For example:

$$\mathbf{W} \sim \text{Uniform} \left( -\sqrt{\frac{6}{\text{fan-in} + \text{fan-out}}}, \sqrt{\frac{6}{\text{fan-in} + \text{fan-out}}} \right)$$

fan-in: # units in previous layer, fan-out: # units in subsequent layer

**Regularization:**  $L_2$  norm penalty, dataset augmentation, injecting noise, dropout (each time a training sample is learned, randomly put hidden units to 0)

**Static vs. dynamic computation graphs:**

*Define-and-Run* vs. *Define-by-Run*

**Batch normalization:** Stabilize the distribution of each layer's activations with mini-batch statistics (apply before non-linearity):

$$\hat{x} = \frac{x - \mathbb{E}_{\mathcal{B}}[x]}{\sqrt{\text{Var}_{\mathcal{B}}[x] + \epsilon}} \text{ and } y = \gamma \hat{x} + \beta \text{ } (\gamma, \beta \text{ learnable})$$

**Skipping Connections:** Improve information flow in DNNs by passing low-level information to higher levels (e.g. with Highway Networks or Residual Connections).

**Tips:**

- Use only differentiable operations
- Always try to overfit the model on a small batch of the training set to make sure that the model is correctly "wired"
- Start with small models and gradually add complexity while monitoring how the performance improves
- Be aware of the properties of activation functions, e.g. no sigmoid output when doing regression
- Monitor the training procedure and use early stopping

## 9 SVM and Kernels

### 9.1 Lagrangian and Dual Function

#### Feasibility:

A point  $\theta \in \mathbb{R}^d$  is called feasible iff it satisfies the constraints of the optimization problem, i.e.  $f_i(\theta) \leq 0 \forall i \in \{1, \dots, M\}$ .

#### Lagrangian:

$$L(\theta, \alpha) = f_0(\theta) + \sum_{i=1}^M \alpha_i f_i(\theta) \quad \text{with } \alpha_i \geq 0$$

Minimize:  $f_0(\theta)$

Subject to:  $f_i(\theta) \leq 0, \quad i = 1, \dots, M$

#### Lagrange dual function / problem:

For every choice of  $\alpha$ , the corresponding unconstrained and concave  $g(\alpha)$  is a lower bound on the optimal value of the constrained problem. Hence,  $\forall \alpha \quad f_0(\theta^*) \geq g(\alpha)$ .

$$g(\alpha) = \min_{\theta \in \mathbb{R}^d} L(\theta, \alpha) = \min_{\theta \in \mathbb{R}^d} \left( f_0(\theta) + \sum_{i=1}^M \alpha_i f_i(\theta) \right)$$

Maximize:  $g(\alpha)$

Subject to:  $\alpha_i \geq 0, \quad i = 1, \dots, m$

Weak Duality:  $d^* \leq p^*$  (Duality Gap:  $p^* - d^* \geq 0$ )

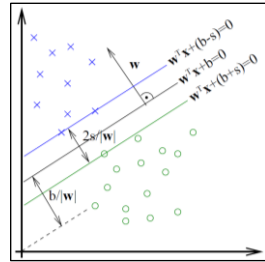
Strong Duality:  $d^* = p^*$ , holds for SVM due to complementary slackness condition  $\alpha_i^* f_i(\theta^*) = 0$ .

$[p^* = f_0(\theta^*), d^* = g(\alpha^*)]$

### 9.2 Hard Margin SVM

#### Goal:

Find a hyperplane that separates both classes with the maximum margin  $m = \frac{2}{\|w\|}$ .  $y_i \in \{-1, 1\}$



**Recipe:** 1) Calculate Lagrangian. 2) Minimize  $L(w, b, \alpha)$  w.r.t.  $w$  and  $b$  (and  $\xi$ ). 3) Formulate ( $\rightarrow$  substitute both relations back into  $L(w, b, \alpha)$ ) and solve dual problem.

#### Objective:

Minimize:  $f_0(w, b) = \frac{1}{2} w^T w = \frac{1}{2} \|w\|^2$

Subject to:  $f_i(w, b) = y_i(w^T x_i + b) - 1 \geq 0$ ,  
for  $i = 1, \dots, N$

#### Primal problem (Lagrangian):

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^N \alpha_i [y_i(w^T x_i + b) - 1]$$

$$\nabla_w L(w, b, \alpha) = w - \sum_{i=1}^N \alpha_i y_i x_i = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^N \alpha_i y_i = 0$$

#### Dual problem:

Maximize:

$$g(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j x_i^T x_j$$

Subject to:

$$\sum_{i=1}^N \alpha_i y_i = 0$$

$$\alpha_i \geq 0, \quad \text{for } i = 1, \dots, N$$

Solving dual problems of SVM is a quadratic problem:

$$g(\alpha) = \frac{1}{2} \alpha^T Q \alpha + \alpha^T \mathbf{1}_N \quad \text{with } Q = -yy^T \odot XX^T \text{ (NSD)}$$

#### Solution:

$$w = \sum_{i=1}^N \alpha_i^* y_i x_i$$

$b = y_i - w^T x_i$  (for  $i$  with  $\alpha_i^* \neq 0$ )  $\rightarrow$  follows from complementary slackness condition  $\alpha_i^* f_i(\theta^*) = 0$ .

#### Classifying:

$$h(x) = \text{sign}(w^T x + b) = \text{sign}\left(\sum_{i=1}^N \alpha_i y_i x_i^T x + b\right)$$

$$\text{sign}(z) = \begin{cases} -1, & \text{if } z < 0 \\ 0, & \text{if } z = 0 \\ 1, & \text{if } z > 0 \end{cases}$$

#### Support vectors:

Due to the complementary slackness condition only samples  $x_i$  which lie on the margin contribute to the weight vector  $w$  (because only for them:  $\alpha_i^* \neq 0, f_i(\theta^*) = 0$ ). These samples are called support vectors.

### 9.3 Soft Margin SVM

**Idea:** Relax the constraints to be able to find a solution for not linearly separable data.  $\rightarrow$  Introduce slack var.  $\xi$

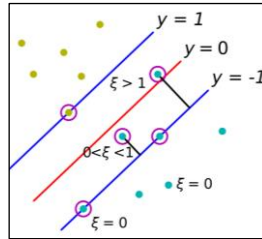
**Objective:** (for  $C \rightarrow \infty$ : Hard Margin SVM)

Minimize:  $f_0(w, b) = \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i, C > 0$

Subject to:  $f_i(w, b) = y_i(w^T x_i + b) - 1 + \xi_i \geq 0$   
 $\xi_i \geq 0, \quad \text{for } i = 1, \dots, N$

#### Primal problem (Lagrangian):

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i [y_i(w^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^N \mu_i \xi_i$$



$$\nabla_w L(w, b, \xi, \alpha, \mu) = w - \sum_{i=1}^N \alpha_i y_i x_i = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^N \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad \text{for } i = 1, \dots, N$$

From  $\alpha_i = C - \mu_i$  and dual feasibility  $\mu_i \geq 0, \alpha_i \geq 0$  we get  $0 \leq \alpha_i \leq C$ .

#### Dual problem:

Maximize:  $\partial_{\alpha_k} g(\alpha) = 1 - \frac{1}{2} \sum_{i=1}^N y_i y_k \alpha_i \alpha_k x_i^T x_k$

$$g(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j x_i^T x_j$$

Subject to:

$$\sum_{i=1}^N \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \dots, N$$

$\alpha_i$  is bounded and cannot go to infinity.

#### Support vectors:

Set of support vectors  $\mathcal{S}$ : points  $x_j$  with  $0 < \alpha_j \leq C$  (all samples on and in the margin and all misclassified samples).

## 9.4 Hinge Loss Formulation (linear soft margin SVM)

Hinge Loss penalizes the points that lie within the margin.

**Objective:** Same as for Soft Margin SVM

**Optimal solution:** Margin violated / not violated

$$\xi_i = \begin{cases} 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) & \text{if } y_i(\mathbf{w}^T \mathbf{x}_i + b) < 1 \\ 0 & \text{else} \end{cases}$$

**Hinge Loss Formulation:**  $E_{\text{hinge}}(z) = \max(0, 1 - z)$

Can be optimized using standard gradient descent.

$$\min_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \underbrace{\sum_{i=1}^N \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))}_{h(\mathbf{w}, b)}$$

$h(\mathbf{w}, b) = 0$  iff hyperplane defined by  $\mathbf{w}$  and  $b$  assigns the correct labels to all samples such that all of them also lie outside of the margin.

## 9.5 Kernels

**Basis functions:**

Use basis functions  $\phi(\mathbf{x}_j)$  with  $\phi: \mathbb{R}^D \rightarrow \mathbb{R}^M$  to make models nonlinear.

**Kernel trick:**

In the dual formulation of SVM, the samples  $\mathbf{x}_i$  only enter the dual objective as inner products  $\mathbf{x}_i^T \mathbf{x}_j$ . Hence, we can replace  $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$  with a kernel function  $k$ .

**Kernel function:**

Encodes similarity between arbitrary numerical or non-numerical data. High outcome  $\rightarrow$  high similarity.

$$k(\mathbf{x}_i, \mathbf{x}_j) := \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \text{ with } k: \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$$

**Validity of kernel:**

A kernel  $k$  is valid, if it corresponds to an inner product in some feature space ( $\phi$ : feature map):

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$$

By the positive definiteness property of the inner product it must hold that  $k(\mathbf{x}, \mathbf{x}) = \phi(\mathbf{x})^T \phi(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathbb{R}^D$ .

**Mercer's theorem:**

A kernel is valid iff its kernel matrix  $\mathbf{K} \in \mathbb{R}^{N \times N}$  is symmetric, positive semidefinite for any input data  $\mathbf{X}$ .

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_N) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & k(\mathbf{x}_N, \mathbf{x}_2) & \dots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

$\mathbf{K}$  is PSD iff  $\det(\mathbf{K}) \geq 0$ .

**Kernel preserving operations:**

Let  $k_1: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  and  $k_2: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  be kernels with  $\mathcal{X} \subseteq \mathbb{R}^N$ . Then the following functions  $k$  are also kernels:

- 1)  $k(\mathbf{x}_1, \mathbf{x}_2) = k_1(\mathbf{x}_1, \mathbf{x}_2) + k_2(\mathbf{x}_1, \mathbf{x}_2)$  [iterated application]
- 2)  $k(\mathbf{x}_1, \mathbf{x}_2) = c \cdot k_1(\mathbf{x}_1, \mathbf{x}_2)$ , with  $c > 0$
- 3)  $k(\mathbf{x}_1, \mathbf{x}_2) = k_1(\mathbf{x}_1, \mathbf{x}_2) \cdot k_2(\mathbf{x}_1, \mathbf{x}_2)$  [iterated application]
- 4)  $k(\mathbf{x}_1, \mathbf{x}_2) = k_3(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2))$ ,  
with the kernel  $k_3$  on  $\mathcal{X}' \subseteq \mathbb{R}^M$  and  $\phi: \mathcal{X} \rightarrow \mathcal{X}'$
- 5)  $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{A} \mathbf{x}_2$ , with  $\mathbf{A} \in \mathbb{R}^{N \times N}$   
symmetric and PSD

Non-valid kernels loose convexity!

**Examples:**

- $\exp\left(-\frac{\mathbf{x}_1^T \mathbf{x}_1}{2\sigma^2}\right) \exp\left(-\frac{\mathbf{x}_2^T \mathbf{x}_2}{2\sigma^2}\right)$ : kernel by rule 4 with  $k_3(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2)$  and  $\phi(\mathbf{z}) = \exp\left(-\frac{\mathbf{z}^T \mathbf{z}}{2\sigma^2}\right)$
- The constant 1 is a kernel by rule 4 with  $k_3(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2)$  and  $\phi(\mathbf{z}) = 1$ .
- Linear kernel:  $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$  (inner product of the input vectors)
- $k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2 + 1)^2 = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2)$  with feature map  $\phi: \mathbb{R}^D \rightarrow \mathbb{R}^M$  with  $M = 1 + D + D^2$ ,  $\phi(\mathbf{x}) = (1 \quad \sqrt{2}x_1 \quad \dots \quad \sqrt{2}x_D \quad x_1x_1 \quad \dots \quad x_1x_D \quad x_2x_1 \quad \dots \quad x_Dx_D)^T$
- $\exp(k(\mathbf{x}_1, \mathbf{x}_2))$  is a kernel if  $k(\mathbf{x}_1, \mathbf{x}_2)$  is a kernel [exercise sheet 09, problem 4b].

**Kernel types:**

Polynomial:  $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T \mathbf{b})^p$  or  $(\mathbf{a}^T \mathbf{b} + 1)^p$

Gaussian:  $k(\mathbf{a}, \mathbf{b}) = \exp\left(-\frac{\|\mathbf{a} - \mathbf{b}\|^2}{2\sigma^2}\right)$

Sigmoid:  $k(\mathbf{a}, \mathbf{b}) = \tanh(\kappa \mathbf{a}^T \mathbf{b} - \delta)$ ,  $\kappa, \delta > 0$   
(not PSD, but still works well in practice)

All finite sets of points can be linearly separated using the Gaussian kernel if the variance  $\sigma$  is chosen small enough [exercise sheet 09, problem 4c].

**Classifying a new point with kernelized SVM:**

[With Soft Margin SVM] From complementary slackness condition  $\alpha_i^* f_i(\theta^*) = 0$  for points  $\mathbf{x}_i \in \mathcal{S}$  with  $\xi_i = 0$ :

$$b = y_i - \mathbf{w}^T \phi(\mathbf{x}_i) = y_i - \sum_{\{j | \mathbf{x}_j \in \mathcal{S}\}} \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

Behaves like kNN-classifier with distance measure  $k(\cdot)$ :

$$h(\mathbf{x}) = \text{sign}\left(\sum_{\{j | \mathbf{x}_j \in \mathcal{S}\}} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}) + b\right)$$

Set of support vectors  $\mathcal{S}$ : points  $\mathbf{x}_j$  with  $0 < \alpha_j \leq C$

**Multiple classes:**

One-vs-Rest: Train  $C$  SVM models for  $C$  classes for One-vs-Rest classification. Winner is the class, where the distance from the hyperplane is maximal.

One-vs-One: Train all possible pairings and evaluate all. The winner is the class with the weighted majority vote.

# 10 Dimensionality Reduction

## 10.1 Principal Component Analysis

The goal of PCA is that we transform the data, such that the covariance between the new dimensions is 0. The dimensions with low variance can then be ignored. → Only captures linear relationships (one solution: Kernel PCA).

**1. Center the data by shifting the mean:**  $X \in \mathbb{R}^{N \times d}$

$$\tilde{x}_i = x_i - \bar{x} \quad \text{with} \quad \bar{x} = \frac{1}{N} X^T \mathbf{1}_N = \frac{1}{N} \sum_{i=1}^N x_i, \quad x \in \mathbb{R}^d$$

$$\tilde{X} = X - \mathbf{1}_N \bar{x}$$

**2. Compute the covariance matrix:**  $\Sigma_X \in \mathbb{R}^{d \times d}$

$$\text{Var}(X_j) = \frac{1}{N} \sum_{i=1}^N (x_{ij} - \bar{x}_j)^2 = \frac{1}{N} X_j^T X_j - \bar{x}_j^2$$

$$\begin{aligned} \text{Cov}(X_{j_1}, X_{j_2}) &= \frac{1}{N} \sum_{i=1}^N (x_{ij_1} - \bar{x}_{j_1})(x_{ij_2} - \bar{x}_{j_2}) \\ &= \frac{1}{N} X_{j_1}^T X_{j_2} - \bar{x}_{j_1} \bar{x}_{j_2} \end{aligned}$$

$$\begin{aligned} \Sigma_X &= \begin{bmatrix} \text{var}(X_1) & \text{cov}(X_1, X_2) & \cdots & \text{cov}(X_1, X_d) \\ \text{cov}(X_2, X_1) & \text{var}(X_2) & \cdots & \text{cov}(X_2, X_d) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(X_d, X_1) & \text{cov}(X_d, X_2) & \cdots & \text{var}(X_d) \end{bmatrix} = \\ &= \frac{1}{N} X^T X - \bar{x} \bar{x}^T = \frac{1}{N} \tilde{X}^T \tilde{X} \\ &= \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^T \end{aligned}$$

**3. Eigenvector decomposition on  $\Sigma_X$ :**

$$\Sigma_X = \Gamma \cdot \Lambda \cdot \Gamma^T \quad \Gamma, \Lambda \in \mathbb{R}^{d \times d}, \Gamma^T = \Gamma^{-1}$$

$\Gamma$ : Orthonormal matrix with columns = normalized eigenvectors  $\gamma_i$  of  $\Sigma_X$  (= principal components).

$\Lambda$ : Diagonal matrix with eigenvalues  $\lambda_i$  of  $\Sigma_X$ . Covariance matrix of new coordinate system.

**4. Transform original data:**

$$Y = \tilde{X} \cdot \Gamma, \quad Y \in \mathbb{R}^{N \times d} \quad \text{reconstruct mean: } \bar{y}^T = \bar{x}^T \cdot \Gamma$$

**Dimensionality reduction:**

Keep only columns (i.e. Eigenvectors) of  $\Gamma$  corresponding to the largest  $k$  Eigenvalues  $\lambda_1, \dots, \lambda_k$  of  $\Sigma_X$ .

$$Y_{\text{reduced}} = \tilde{X} \cdot \Gamma_{\text{tr}} \quad \Gamma_{\text{tr}} \in \mathbb{R}^{d \times k}, Y_{\text{reduced}} \in \mathbb{R}^{N \times k}$$

$$90\% \text{ rule: } \sum_{i=1}^k \lambda_i \geq 0.9 \cdot \sum_{i=1}^d \lambda_i$$

**Power iteration:**

Iterative approach to compute the eigenvector with the largest absolute value. Initialize with arbitrary normalized vector  $v$  and iteratively compute until convergence:

$$v \leftarrow \frac{\Sigma_X \cdot v}{\|\Sigma_X \cdot v\|}$$

Deflated Matrix:  $\hat{\Sigma}_X = \Sigma_X - \lambda_1 \cdot \gamma_1 \gamma_1^T \rightarrow$  1st EV becomes 0

**Maximum variance formulation:**

Project the data to a lower dimensional space  $\mathbb{R}^k$  with  $k < d$  while maximizing the variance of the projected data.

**Minimum error formulation:**

Find an orthonormal set of  $k$  basis vectors  $u_i \in \mathbb{R}^d$  and corresponding low-dimensional projections  $z_{ni}$  for every sample  $x_n \in \mathbb{R}^d$  such that the average reconstruction error is minimized:  $\hat{x}_n = U z_n + \mu$ ,  $z_n \in \mathbb{R}^k$ ,  $U \in \mathbb{R}^{d \times k}$ .

## 10.2 Singular Value Decomposition

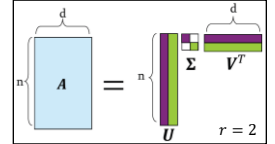
The goal of SVD is to find the best low rank approximation (regarding Frobenius norm) by minimizing the reconstruction error.

$$\min_{\text{rank}(B)=k} \|X - B\|_F^2 = \min_{\text{rank}(B)=k} \sum_{i=1}^N \sum_{j=1}^d (x_{ij} - b_{ij})^2$$

**Matrix decomposition:**  $r = \text{rank}(X) \leq \min(d, N)$

$$X = U \cdot \Sigma \cdot V^T = \sum_{i=1}^r \sigma_i u_i \cdot v_i^T$$

$$X \in \mathbb{R}^{N \times d}, u_i \in \mathbb{R}^{N \times 1}, v_i^T \in \mathbb{R}^{1 \times d}$$



$U \in \mathbb{R}^{N \times r}$ : orthonormal columns  $u_i \rightarrow U^T U = I$

User/Entry-to-concept similarity

$\Sigma \in \mathbb{R}^{r \times r}$ : Diagonal matrix with singular values  $\sigma_i > 0$  in decreasing order

Strength of each concept

$V \in \mathbb{R}^{d \times r}$ : orthonormal columns  $v_i \rightarrow V^T V = I$

$v_i$ : new coordinate axes

Feature-to-concept similarity

**Projected data** (Coordinates in new reference frame):

$$X_{\text{proj}} = X \cdot V = U \cdot \Sigma$$

**Best low rank approximation / Truncated SVD:**

$$B = U \cdot \Sigma_{\text{tr}} \cdot V^T = U_{\text{tr}} \cdot \Sigma_{\text{tr}} \cdot V_{\text{tr}}^T = \sum_{i=1}^k \sigma_i u_i \cdot v_i^T$$

$\Sigma_{\text{tr}} \in \mathbb{R}^{r \times r}$  with  $\sigma_{\text{tr},i} = \sigma_i$  for  $i = 1 \dots k$  and  $\sigma_{\text{tr},i} = 0$  else.

$$X_{\text{proj}} = U_{\text{tr}} \cdot \Sigma_{\text{tr}} = B \cdot V_{\text{tr}} = X \cdot V_{\text{tr}}$$

$V_{\text{trunc}} \in \mathbb{R}^{d \times r}$ :  $v_{\text{tr},i} = v_i$  for  $i = 1 \dots k$  and  $v_{\text{tr},i} = 0$  else.

$U_{\text{trunc}} \in \mathbb{R}^{N \times r}$ :  $u_{\text{tr},i} = u_i$  for  $i = 1 \dots k$  and  $u_{\text{tr},i} = 0$  else.

$$90\% \text{ rule: } \sum_{i=1}^k \sigma_i^2 \geq 0.9 \cdot \sum_{i=1}^d \sigma_i^2$$

**SVD vs. PCA:**

Transform the data such that dimensions of new space are uncorrelated and discard new dimensions with smallest variance (PCAs) is equivalent to finding the optimal low-rank approximation regarding the Frobenius norm (SVD).

$$\frac{1}{N} \tilde{X}^T \tilde{X} = \frac{1}{N} V \Sigma^2 V^T \stackrel{!}{=} \Sigma_{\tilde{X}} = \Gamma \Lambda \Gamma^T \rightarrow V = \Gamma, \quad \frac{\Sigma^2}{N} = \Lambda$$

**Compute SVD:**

- $X^T X = V \Sigma^2 V^T \rightarrow V$  = Eigenvectors of  $X^T X$
- $XX^T = U \Sigma^2 U^T \rightarrow U$  = Eigenvectors of  $XX^T$
- Get  $\Sigma$  by taking the square roots of the Eigenvalues of  $X^T X$  or  $XX^T$ .

**Frobenius norm formulas:**

$$\|X\|_F = \|X^T\|_F \quad \|X\|_F^2 = \text{tr}(X^T X) \quad \|X\|_F^2 = \sum_{i,j} X_{ij}^2$$

$$\text{With } X = U \Sigma V^T: \|X\|_F^2 = \|\Sigma\|_F^2 = \sum_{i=1}^r \sigma_i^2$$



# 11 Dim. Red. & Matrix Factorization

## 11.1 Latent Factor Model

Motivation: Recommender systems (4 cold start problem)

### 11.1.1 Basics

**RMSE for evaluating recommender systems:**

$$RMSE = \frac{1}{|S|} \sqrt{\sum_{(u,i) \in S} (r_{ui} - \hat{r}_{ui})^2}$$

$S$ : Set of tuples  $(u, i)$  of users  $u$  that rated item  $i$  with a rating of  $r_{ui}$

**SVD on rating data:** → minimizes RMSE

Standard SVD cannot be used because missing elements are treated as zeros and lack of sparsity.  $\mathbf{R} \in \mathbb{R}^{n \times d}$

$$\mathbf{R} \approx \mathbf{Q} \cdot \mathbf{P}^T \text{ with } \mathbf{Q} = \mathbf{U}\mathbf{\Sigma} \in \mathbb{R}^{n \times k} \text{ and } \mathbf{P} = \mathbf{V} \in \mathbb{R}^{d \times k}$$

**Latent factor minimization problem:**

We only sum over existing entries and do not require  $\mathbf{Q}, \mathbf{P}$  to be orthogonal and unit length.  $\mathbf{q}_u \in \mathbb{R}^{1 \times k}, \mathbf{p}_i \in \mathbb{R}^{1 \times k}$

$k$ : number of latent dimensions ( $k \leq \text{rank}(\mathbf{R})$ )

$$\min_{\mathbf{P}, \mathbf{Q}} \sum_{(u,i) \in S} (r_{ui} - \mathbf{q}_u \cdot \mathbf{p}_i^T)^2 \quad \begin{array}{l} \hat{r}_{ui} = \mathbf{q}_u \cdot \mathbf{p}_i^T \\ \mathbf{q}_u \text{ row } u \text{ of } \mathbf{Q} \\ \mathbf{p}_i \text{ row } i \text{ of } \mathbf{P} \end{array}$$

### 11.1.2 Alternating Optimization

**Initialization:**

- Use SVD where missing entries are replaced by 0
- Random Initialization (e.g. normal-distributed)

**Alternating optimization:**

Alternatingly keep one variable fix and optimize for the other. Repeat until convergence.

$$\begin{aligned} \mathbf{P}^{(t+1)} &= \arg \min_{\mathbf{P}} f(\mathbf{P}, \mathbf{Q}^{(t)}) \\ &= \min_{\mathbf{P}} \sum_{(u,i) \in S} (r_{ui} - \mathbf{q}_u \cdot \mathbf{p}_i^T)^2 + \lambda_2 \|\mathbf{p}_i\|^2 \\ &= \sum_{i=1, \dots, d} \min_{\mathbf{p}_i} \sum_{u \in S_{*,i}} (r_{ui} - \mathbf{q}_u \cdot \mathbf{p}_i^T)^2 + \lambda_2 \|\mathbf{p}_i\|^2 \end{aligned}$$

Where  $S_{*,i} = \{u | (u, i) \in S\}$  (only users who have rated item  $i$ )

$$\begin{aligned} \mathbf{Q}^{(t+1)} &= \arg \min_{\mathbf{Q}} f(\mathbf{P}^{(t+1)}, \mathbf{Q}) \\ &= \min_{\mathbf{Q}} \sum_{(u,i) \in S} (r_{ui} - \mathbf{q}_u \cdot \mathbf{p}_i^T)^2 + \lambda_1 \|\mathbf{q}_u\|^2 \\ &= \sum_{u=1, \dots, n} \min_{\mathbf{q}_u} \sum_{i \in S_{u,*}} (r_{ui} - \mathbf{q}_u \cdot \mathbf{p}_i^T)^2 + \lambda_1 \|\mathbf{q}_u\|^2 \end{aligned}$$

Where  $S_{u,*} = \{i | (u, i) \in S\}$  (only items that have been rated by user  $u$ )

**Closed form solution:**  $\mathbf{w} \triangleq \mathbf{p}_i^T, \mathbf{x}_j \triangleq \mathbf{q}_u^T, y_j \triangleq r_{ui}$  (1<sup>st</sup> equ.)

Ordinary least squares problem:  $L_2$ -Regularization

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_d)^{-1} \mathbf{X}^T \mathbf{y} = \left( \frac{1}{n} \sum_{j=1}^n \mathbf{x}_j \mathbf{x}_j^T + \lambda \mathbf{I}_d \right)^{-1} \cdot \frac{1}{n} \sum_{j=1}^n \mathbf{x}_j y_j$$

$$\mathbf{p}_i^T = \left( \frac{1}{|S_{*,i}|} \sum_{u \in S_{*,i}} \mathbf{q}_u^T \mathbf{q}_u + \lambda \cdot \mathbf{I}_d \right)^{-1} \cdot \frac{1}{|S_{*,i}|} \sum_{u \in S_{*,i}} \mathbf{q}_u^T r_{ui}$$

$$\mathbf{q}_u^T = \left( \frac{1}{|S_{u,*}|} \sum_{i \in S_{u,*}} \mathbf{p}_i^T \mathbf{p}_i + \lambda \cdot \mathbf{I}_d \right)^{-1} \cdot \frac{1}{|S_{u,*}|} \sum_{i \in S_{u,*}} \mathbf{p}_i^T r_{ui}$$

Solution is only an approximation and depends on the initialization.

**User and item bias:**

$$\min_{\mathbf{P}, \mathbf{Q}} \sum_{(u,i) \in S} (r_{ui} - (\mathbf{q}_u \cdot \mathbf{p}_i^T + b_u + b_i + b))^2$$

**Predictions:**

$$\hat{r}_{ui} = \mathbf{q}_u \cdot \mathbf{p}_i^T$$

### 11.1.3 SGD for Matrix Factorization

- Pick random user  $u$  and a random item  $i$  with rating  $r_{ui}$  (batch size 1)
- Compute the gradients w.r.t.  $\frac{\partial \mathcal{L}}{\partial \mathbf{q}_u}$  and  $\frac{\partial \mathcal{L}}{\partial \mathbf{p}_i}$
- Update Rule:  $\mathbf{q}_u \leftarrow \mathbf{q}_u - \tau \frac{\partial \mathcal{L}}{\partial \mathbf{q}_u}, \mathbf{p}_i \leftarrow \mathbf{p}_i - \tau \frac{\partial \mathcal{L}}{\partial \mathbf{p}_i}$

Objective Function:

$$\mathcal{L} = \sum_{(u,i) \in S} (r_{ui} - \mathbf{q}_u \cdot \mathbf{p}_i^T)^2 + \text{Bias} + \text{Regularization}$$

$$e_{ui} \leftarrow r_{ui} - (\mathbf{q}_u \cdot \mathbf{p}_i^T + b_u + b_i + b)$$

$$\mathbf{q}_u \leftarrow \mathbf{q}_u + 2\tau(e_{ui} \mathbf{p}_i - \lambda_1 \mathbf{q}_u)$$

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + 2\tau(e_{ui} \mathbf{q}_u - \lambda_2 \mathbf{p}_i)$$

$$b_u \leftarrow b_u + 2\tau e_{ui}$$

$$b_i \leftarrow b_i + 2\tau e_{ui}$$

$$b = \frac{1}{|S|} \sum_{(u,i) \in S} r_{ui}$$

### 11.1.4 Regularization

$$\min_{\mathbf{P}, \mathbf{Q}} \sum_{(u,i) \in S} (r_{ui} - \mathbf{q}_u \cdot \mathbf{p}_i^T)^2 + \left[ \lambda_1 \sum_u \|\mathbf{q}_u\|^2 + \lambda_2 \sum_i \|\mathbf{p}_i\|^2 \right]$$

Lots of available ratings → error term dominates

Few ratings available → regularization term is more dominant

**L2 regularization:**

- Tries to shrink the parameter vector  $\mathbf{w}$  equally.
- Large values are highly penalized due to the square.
- It is unlikely that any component will be exactly 0.

**L1 regularization:**

- allows large values in parameter vector  $\mathbf{w}$  by shrinking other components to 0
- is suited to enforce sparsity of the parameter vector

### 11.1.5 Non-Negative Matrix Factorization

Factorize non-negative  $\mathbf{A}$  in non-negative  $\mathbf{Q}$  and  $\mathbf{P}$ , i.e.

$$\mathbf{A} \approx \mathbf{Q} \cdot \mathbf{P}^T. \text{ Constrained minimization problem:}$$

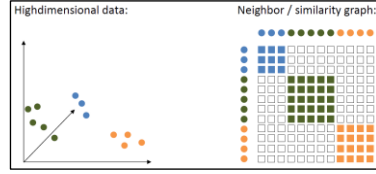
$$\min_{\mathbf{P} \geq 0, \mathbf{Q} \geq 0} \|\mathbf{A} - \mathbf{Q} \cdot \mathbf{P}^T\|_F \quad \mathbf{Q} \in \mathbb{R}^{n \times k}, \mathbf{P} \in \mathbb{R}^{d \times k}, \mathbf{A} \in \mathbb{R}^{n \times d}$$

## 11.2 Neighbor Graph Methods

**Idea:** Preserve local similarity (vs. global similarity in PCA) of high-dimensional data in low-dim. approximation.

### General Approach:

1. Construct neighbor graph of high-dim. data
2. Initialize points randomly in low-dim. space
3. Optimize coordinates in low-dim space s.t. similarities align



### 11.2.1 t-Distributed Stochastic Neighbor Embedding (t-SNE)

Non-linear dimensionality reduction.

**High-dimensional similarities for input  $x_i$ :**

$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)} \quad p_{ii} = 0$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Choose  $\sigma_i$  such that perplexity (effective number of neighbors) is constant ( $\sim 5 - 15$ ).

**Low dimensional similarities for parameters  $y_i$ :** (to be optimized)

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}} \quad q_{ii} = 0$$

**Minimize the KL divergence:**

$$\min_{y_i} \text{KL}(\mathbf{P}||\mathbf{Q}) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$\frac{\partial \text{KL}(\mathbf{P}||\mathbf{Q})}{\partial y_s} = 4 \sum_j (y_s - y_j)(p_{sj} - q_{sj})(1 + \|y_s - y_j\|^2)^{-1}$$

## 11.3 Autoencoders

Neural network that finds a compact representation of non-linear data by learning to reconstruct its input:  $f(\mathbf{x}, \mathbf{W}) = \hat{\mathbf{x}} \approx \mathbf{x}$ .

**Objective:**

$$\min_{\mathbf{W}} \frac{1}{N} \sum_{i=1}^N \|f(\mathbf{x}_i, \mathbf{W}) - \mathbf{x}_i\|^2$$

**Alternative view:**

Find a latent representation / code  $\mathbf{z} \in \mathbb{R}^L$  which is a compact representation for  $\mathbf{x} \in \mathbb{R}^D$  since  $L \ll D$ .

$f_{\text{enc}}(\mathbf{x}) = \mathbf{z}$  **Encoder:** project data to a lower dim.

$f_{\text{dec}}(\mathbf{z}) \approx \mathbf{x}$  **Decoder:** reconstruct the data

$$f(\mathbf{x}) = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x})) \approx \mathbf{x}$$

The middle layer (bottleneck layer) has fewer neurons ( $L \ll D$ ). When using Autoencoder with linear activation functions and optimal weights, it is equivalent to SVD and PCA with the optimal solution (assuming normalization):

$$\mathbf{W}^* = \mathbf{\Gamma}_{\text{tr}} \text{ where } \mathbf{\Gamma}_{\text{tr}} \text{ are the top-}L \text{ eigenvectors of } \mathbf{X}^T \mathbf{X}.$$

## 12 Clustering

Group objects  $x_i$  into  $K$  groups (clusters) based on their similarity.

### 12.1 K-Means

**Distance measures  $d(x_i, x_j)$ :**

$$\text{Manhattan } (L_1): \|x_i - x_j\|_1 = \sum_d |x_{id} - x_{jd}|$$

$$\text{Euclidean } (L_2): \|x_i - x_j\|_2 = \sqrt{\sum_d (x_{id} - x_{jd})^2}$$

$$\text{Mahalanobis: } \sqrt{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)}$$

**Objective Function / Distortion Measure:**

$$J(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}) = \sum_{i=1}^N \sum_{k=1}^K z_{ik} \|x_i - \mu_k\|_2^2$$

with one-hot encoding  $\mathbf{z}_i \in \{0,1\}^K$  and  $\boldsymbol{\mu}_k \in \mathbb{R}^D$

**Goal:**

$$\mathbf{Z}^*, \boldsymbol{\mu}^* = \underset{\mathbf{Z}, \boldsymbol{\mu}}{\text{argmin}} J(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}) \quad \mathbf{Z}^* \in \mathbb{R}^{N \times K}$$

**Alternating optimization with Lloyd's algorithm:**

1. Initialize the centroids  $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$
2. Update cluster indicators (solve  $\min_{\mathbf{Z}} J(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu})$ )

$$z_{ik} = \begin{cases} 1, & \text{if } k = \underset{j}{\text{argmin}} \|x_i - \mu_j\|_2^2 \\ 0, & \text{else} \end{cases}$$

3. Update centroids (solve  $\min_{\boldsymbol{\mu}} J(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu})$ )

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^N z_{ik} \mathbf{x}_i \quad \text{where} \quad N_k = \sum_{i=1}^N z_{ik}$$

For  $L_1$ -norm:  $\mu_{kd} = \text{median}(\{x_{id}: i = 1, \dots, N \text{ and } z_{ik} = 1\})$

4. If objective  $J(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu})$  has not converged  $\rightarrow$  Step 2

**Initialization of centroids with K-means++:**

1. Choose the first centroid  $\boldsymbol{\mu}_1$  uniformly at random among the data points.
2. For each point  $x_i$  compute the distance  $D_i^2 = \|x_i - \boldsymbol{\mu}_1\|_2^2$ .
3. Sample the next centroid  $\boldsymbol{\mu}_k$  from  $\{x_i\}$  with probability proportional to  $D_i^2$ .
4. Recompute the distances.  
 $D_i^2 = \min\{\|x_i - \boldsymbol{\mu}_1\|_2^2, \dots, \|x_i - \boldsymbol{\mu}_k\|_2^2\}$
5. Continue steps 3 and 4 until  $K$  initial centroids have been chosen.

**Limitations:**

**Modeling Issues:**  $\rightarrow$  distortion  $J(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu})$

- Underlying assumptions are not explicit
- Can't detect clusters w/ overlapping convex hulls
- Sensitivity to outliers
- No uncertainty measure

**Algorithmic Issues:**  $\rightarrow$  Lloyd's algorithm

- Extreme sensitivity to initialization

## 12.2 Gaussian Mixture Models (GMMs)

Fit set of Gaussians to the data (via EM). Cluster is the Gaussian with the highest probability.

Learning:  $\pi^*, \mu^*, \Sigma^* = \underset{\pi, \mu, \Sigma}{\operatorname{argmax}} \log p(X|\pi, \mu, \Sigma)$

Inference:  $p(Z|X, \pi, \mu, \Sigma)$

**Model:**  $\theta = \{\pi, \mu, \Sigma\}$  (joint log-likelihood)

$P(X, Z|\theta) = \prod_i p(x_i, z_i|\theta)$  (i.i.d.)

$p(x_i, z_i|\theta) = p(x_i|z_i, \theta) \cdot p(z_i|\theta)$

Cluster Prior:  $p(z_i|\theta) = \text{Cat}(\pi)$

Multivariate normal distribution per cluster:

$p(x_i|z_{ik} = 1, \theta) = \mathcal{N}(x_i|\mu_k, \Sigma_k)$

**Likelihood:**

$p(x_i|\pi, \mu, \Sigma) = \sum_{k=1}^K p(z_{ik} = 1|\pi) \cdot p(x_i|z_{ik} = 1, \mu_k, \Sigma_k)$   
 $= \sum_{k=1}^K \pi_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k)$

$\log p(X|\pi, \mu, \Sigma) = \log \prod_{i=1}^N p(x_i|\pi, \mu, \Sigma)$   
 $= \sum_{i=1}^N \log \sum_{k=1}^K p(x_i|z_k, \mu_k, \Sigma_k) \cdot p(z_k|\pi_k)$   
 $= \sum_{i=1}^N \log \sum_{k=1}^K \pi_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k)$

**Inference: Posterior / Responsibility:**  $z_{ik} = 1 \triangleq z_i = k$

$\gamma(z_{ik}) = p(z_{ik} = 1|x_i, \pi, \mu, \Sigma)$   
 $= \frac{p(z_{ik} = 1|\pi) \cdot p(x_i|z_{ik} = 1, \mu_k, \Sigma_k)}{p(x_i|\pi, \mu, \Sigma)}$   
 $= \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$

**EM Algorithm for GMM:**

1. Initialize model parameters  $\{\pi^{(0)}, \mu_1^{(0)}, \dots, \mu_K^{(0)}, \Sigma_1^{(0)}, \dots, \Sigma_K^{(0)}\}$ .
2. **E step.** Evaluate the responsibilities (assume  $\gamma_t(Z) = \sum_{i=1}^N \gamma_t(z_{ik})$ )

$$\gamma_t(z_{ik}) = \frac{\pi_k^{(t)} \mathcal{N}(x_i|\mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_{j=1}^K \pi_j^{(t)} \mathcal{N}(x_i|\mu_j^{(t)}, \Sigma_j^{(t)})}$$

3. **M step.** Re-estimate the parameters

$$\pi^{(t+1)}, \mu^{(t+1)}, \Sigma^{(t+1)} = \underset{\pi, \mu, \Sigma}{\operatorname{argmax}} \mathbb{E}_{Z \sim \gamma_t(Z)} [\log p(X, Z|\pi, \mu, \Sigma)]$$

$$\mu_k^{(t+1)} = \frac{1}{N_k} \sum_{i=1}^N \gamma_t(z_{ik}) x_i$$

$$\Sigma_k^{(t+1)} = \frac{1}{N_k} \sum_{i=1}^N \gamma_t(z_{ik}) (x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T$$

$$\pi_k^{(t+1)} = \frac{N_k}{N} \quad \text{where} \quad N_k = \sum_{i=1}^N \gamma_t(z_{ik})$$

4. Iterate steps 2 & 3 until  $\mathbb{E}_{Z \sim \gamma_t(Z)} [\log p(X, Z|\pi^{(t)}, \mu^{(t)}, \Sigma^{(t)})]$  converges.

**Generative process of GMM:**

1. Draw a one-hot cluster indicator  $z \sim \text{Cat}(\pi)$ .  
 $z_k = 1 \Leftrightarrow$  current point belongs to cluster  $k$ .
2. Draw the sample  $x \sim \mathcal{N}(\mu_k, \Sigma_k)$  if  $z_k = 1$ .

**Choosing number of clusters  $K$ : Heuristic methods**

- **Elbow/knee heuristic:** Plot within-cluster sum of squared distances / likelihood for varying  $K$ . Look for a bent.
- **Gap statistic:** Compare within-cluster variation to uniform data. Choose smallest  $K$  such that gap statistic is within one std. dev. of gap at  $K + 1$ .
- **Silhouette:** Per point difference of mean distance within cluster to points in closest other cluster. Maximize mean of all points.

**Choosing number of clusters  $K$ : Probabilistic methods**

Needs generative model that defines the data likelihood  $\hat{L} = p(X|\hat{Z}, \hat{\theta})$  (with optimal parameters  $\hat{Z}, \hat{\theta}$ ).

- **Bayesian information criterion (BIC):** Approximate model likelihood  $p(X|\text{model})$ . Balances number of parameters vs. likelihood,  $\text{BIC} = M \log N - 2 \log \hat{L}$ .
- **Akaike information criterion (AIC):** Estimate information lost by given model,  $\text{AIC} = 2M - 2 \log \hat{L}$ .
- Here  $M$  is the number of free parameters (e.g.  $K \cdot (D + D^2 + 1)$  for GMM) and  $N$  is the number of samples.

## 12.3 Expectation-Maximization Algorithm

Should be used when it is intractable to optimize the log-likelihood  $\log p(X|\theta) = \log(\sum_Z p(X|Z, \theta) \cdot p(Z|\theta))$  due to latent variables  $Z$ . Easier to optimize the joint probability  $\log p(X, Z|\theta) = \log p(Z|\theta) + \log p(X|Z, \theta)$ .

**Iteration scheme:**

- Use our current beliefs  $p(Z|X, \theta^{(t)})$  to “pretend” that we know  $Z$ .
- **E step:** Update the responsibilities  $\gamma_t(Z) = p(Z|X, \theta^{(t)})$
- **M step:** Update parameters

$$\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}} \mathbb{E}_{Z \sim \gamma_t(Z)} [\log p(X, Z|\theta)]$$

$$\begin{aligned} \mathbb{E}_{Z \sim \gamma_t(Z)} [\log p(X, Z|\theta)] &= \sum_Z \gamma_t(Z) \log p(X, Z|\theta) = \\ &= \sum_{i=1}^N \sum_{k=1}^K \gamma_t(z_{ik}) \log p(x_i, z_{ik} = 1|\theta) = \\ &= \sum_{i=1}^N \sum_{k=1}^K \gamma_t(z_{ik}) [\log p(x_i|z_{ik} = 1, \theta) + \log p(z_{ik} = 1|\theta)] \end{aligned}$$

**Justification:**

Goal: Maximize  $\log p(X|\theta) \rightarrow$  optimization intractable

$$\log p(X|\theta) = \mathcal{L}(q, \theta) + \mathbb{KL}(q(Z)||p(Z|X, \theta))$$

$$\mathcal{L}(q, \theta) = \mathbb{E}_{Z \sim q} \left[ \log \frac{p(X, Z|\theta)}{q(Z)} \right]$$

$\mathcal{L}(q, \theta)$  is a lower-bound on  $\log p(X|\theta)$  for any  $q$ , because  $\mathcal{L}(q, \theta)$  is maximal when the  $\mathbb{KL}$  divergence is 0, which happens when  $q(Z) = \gamma(Z) = p(Z|X, \theta)$ .

After the E-step we have  $\log p(X|\theta) = \mathcal{L}(q, \theta)$ . So if we now perform the M-step and maximize  $\mathcal{L}(q, \theta)$  with respect to  $\theta$ , either

- $\mathcal{L}(q, \theta)$  does not change, we are at a local maximum of  $\log p(X|\theta)$  and EM converges
- or  $\mathcal{L}(q, \theta)$  “pushes” up against  $\log p(X|\theta)$ .

## 12.4 Hierarchical Clustering

**Agglomerative:** Bottom-up, merge cluster pairs iteratively (examples for linkage criterions: Complete-linkage clustering, Centroid linkage clustering)

**Divisive:** Top-down, split data recursively