

252-0535-00: Advanced Machine Learning

Conceptual Summary

Swiss Federal Institute of Technology Zurich (ETH Zürich)

Last Edit Date: 02/08/2025

Disclaimer and Term of Use:

1. We do not guarantee the accuracy and completeness of the summary content. Some of the course material may not be included, and some of the content in the summary may not be correct. You should use this file properly and legally. We are not responsible for any results from using this file.
2. This personal note is adapted (derived) from the slides and/or codes from *Prof. Joachim Buhmann and Dr. Carlos Jimenez*. Please [contact us](#) to delete this file if you think your rights have been violated.
3. This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

I. Preliminaries

I. Matrix Calculus

$$\frac{\partial x^T A}{\partial x} = \frac{\partial A^T x}{\partial x} = A, \frac{\partial A^T X A}{\partial x} = A^T (X + X^T), \\ \frac{\partial K^{-1}}{\partial K} - K^{-1} K' K^{-1}, \frac{\partial}{\partial x^2} x^T A x = A + A^T.$$

II. Probability

$P(\emptyset) = 0$; $P(\Omega) = 1$; if $\{A_i\}_{i=1}^\infty \subseteq \Sigma$ are mutually independent, we have $P(U_{i=1}^\infty A_i) = \prod_{i=1}^\infty P(A_i)$. **Bayes' Rule** $p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}$; $p(\mathbf{y}) = \sum_i p(\mathbf{y}|\mathbf{x}_i)p(\mathbf{x}_i)$ marginal likelihood, $p(\mathbf{x}|\mathbf{y})$ posterior, $p(\mathbf{x})$ prior, $p(\mathbf{y}|\mathbf{x})$ conditional likelihood.

III. Expectation & Variance Matrix

$E[\mathbf{A}\mathbf{X} + \mathbf{b}] = \mathbf{A}E[\mathbf{X}] + \mathbf{b}$; $E[\mathbf{X} + \mathbf{Y}] = E[\mathbf{X}] + E[\mathbf{Y}]$; $\text{Var}(\mathbf{A}\mathbf{X} + \mathbf{b}) = \mathbf{A}\text{Var}(\mathbf{X})\mathbf{A}^T + \mathbf{b}$; $\text{Var}(\mathbf{X} + \mathbf{Y}) = \text{Var}(\mathbf{X}) + \text{Var}(\mathbf{Y}) + 2\text{Cov}(\mathbf{X}, \mathbf{Y})$ if independent $\text{Cov} = 0$; $E[\mathbf{X}\mathbf{Y}^T] = E[\mathbf{X}]E[\mathbf{Y}]^T$ if independent; $E[g(\mathbf{X})] = \int g(\mathbf{x})f(\mathbf{x})d\mathbf{x}$ if $\mathbf{X} \in R, g: R^d \rightarrow R$; $E[E[\mathbf{X}|\mathbf{Y}]] = E[\mathbf{X}]$; $\text{Var}(\mathbf{X}) = E[\text{Var}(\mathbf{X}|\mathbf{Y})] + \text{Var}(E[\mathbf{X}|\mathbf{Y}])$; $E(f(\mathbf{x})) \leq f(E(\mathbf{x}))$ **Jensen's inequality** f convex.

IV. Information Theory

Entropy $H(\mathbf{X}) = -\sum P(\mathbf{X} = x) \log P(\mathbf{X} = x)$, $H(\mathbf{X}) = -\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}$ (continuous). $H(\mathbf{X}, \mathbf{Y}) = -\sum P(\mathbf{X} = x|\mathbf{Y} = y) \log P(\mathbf{X} = x|\mathbf{Y} = y)$ **joint entropy**. $H(\mathbf{X}|\mathbf{Y}) = -\sum P(\mathbf{X} = x|\mathbf{Y} = y) \log P(\mathbf{X} = x|\mathbf{Y} = y) = -\sum P(\mathbf{X} = x, \mathbf{Y} = y) \log \frac{P(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y})}{P(\mathbf{Y}=\mathbf{y})}$ **conditional entropy**. $I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) = -\sum P(\mathbf{X} = x, \mathbf{Y} = y) \log \frac{P(\mathbf{X}=\mathbf{x})P(\mathbf{Y}=\mathbf{y})}{P(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y})}$ **mutual information**. $KL(p; q) = \sum p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}$ is KL

divergence; $CE(p|q) = -\sum p(\mathbf{x}) \log q(\mathbf{x}) = KL(p; q) + H(p)$ is **cross entropy**. $* H(\mathbf{X}) \geq 0$; $H(\mathbf{X}, \mathbf{Y}) = H(\mathbf{X}) + H(\mathbf{Y})$ if $\mathbf{X}, \mathbf{Y}$ independent; $0 \leq I(\mathbf{X}; \mathbf{Y}) \leq H(\mathbf{X})$; $I(\mathbf{X}; \mathbf{Y}) = I(\mathbf{Y}; \mathbf{X})$; KL divergence quantifies the difference between the number of bits required for encoding p using a code optimized for q , p .

2. Anomaly Detection

I. Principal Component Analysis (PCA)

PCA is for dimension reduction $R^D \rightarrow R^d$ such that d components have the **largest variance**, larger variance are **more informative**. Suppose $\bar{X} = \frac{1}{n} \sum_{i=1}^n x_i$ is the sample mean, $u \in R^D$, $\|u\|_2^2 = u^T u = 1$, the **sample mean** of $\{u^T x_1, \dots, u^T x_n\}$ is $u^T \bar{X}$, the **sample variance** is $\frac{1}{n} \sum_{i=1}^n (u^T x_i - u^T \bar{X})^2 = u^T \left(\frac{1}{n} \sum_{i=1}^n (x_i - \bar{X})(x_i - \bar{X})^T \right) u = u^T S u$. Then $u_1 = \text{argmax}_{u \in R^D, \|u\|_2=1} u^T S u$, where u is the unit eigenvector of S that corresponds the largest eigenvalue. Let $f(u) = u^T S u$, $g(u) = 1 -$

$u^T u$, maximize $f(u)$ such that $g(u) = 0$. The Lagrangian is $L(u, \lambda) = f(u) + \lambda g(u) = u^T S u + \lambda(1 - u^T u) = u^T (S - \lambda I_D) u + \lambda$, $\frac{\partial L}{\partial u}(u_1, \lambda) = 2(S - \lambda I_D)u_1 = 0$, $S u_1 = \lambda u_1$. We define the first PC as $X_1 = \{x - \text{proj}_{u_1} x: x \in X\} = \{x - (u_1^T x) \cdot u_1: x \in X\}$.

*** Suppose eigenvalue associated with u_1 is λ_1 , then for any unit eigenvector u^* of S associated with eigenvalue λ^* we have $\lambda_1 = \lambda_1 u_1^T u_1 = u_1^T S u_1 \geq u^{*T} S u^* = \lambda^* u^{*T} u^* = \lambda^*$.**

II. Gaussian Mixture Model, EM Algorithm

Steps of GMM: **1)** Randomly pick one distribution from K distinct *Gaussian distributions*; **2)** Generate the data point according to the selected distribution. *GMM pdf* is $p(\mathbf{x}) = \sum_{k=1}^K \pi_k N(\mathbf{x}|\mu_k, \Sigma_k)$, where $\{\pi_k\}_{k=1}^K$ is probability distribution, $N(\mathbf{x}|\mu_k, \Sigma_k)$ is the pdf of distribution $N(\mu_k, \Sigma_K)$. Log-likelihood is $\log p_\theta(\mathbf{X}) = \sum_{i=1}^n \log p_\theta(x_i)$, but intractable. Decompose to $\log p_\theta(\mathbf{X}, \mathbf{Z}) = \sum_{i=1}^n \log \pi_{z_i} + \sum_{i=1}^n \log N(x_i|\mu_{z_i}, \Sigma_{z_i})$.

Maximize $\log p_\theta(\mathbf{X}) = E_{z \sim q} \left[\log \left(\frac{p_\theta(\mathbf{X}, \mathbf{Z})}{p_\theta(\mathbf{Z}|\mathbf{X})} \cdot \frac{q(\mathbf{Z})}{q(\mathbf{Z})} \right) \right] = E_{z \sim q} \left[\frac{p_\theta(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] + E_{z \sim q} \left[\log \left(\frac{q(\mathbf{Z})}{p_\theta(\mathbf{Z}|\mathbf{X})} \right) \right]$, we denote $M(q, \theta) = E_{z \sim q} \left[\log \left(\frac{p_\theta(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right) \right]$, $E(q, \theta) = E_{z \sim q} \left[\log \left(\frac{q(\mathbf{Z})}{p_\theta(\mathbf{Z}|\mathbf{X})} \right) \right]$. **Optimization steps:** **1)** Initialize θ_0 ; **2)** For $t = 1, \dots$, update $q^t = \text{argmin}_q E(q, \theta^{t-1}) = p_{\theta^{t-1}}(\cdot | \mathbf{X})$, $\theta^t = \text{argmax}_\theta M(q^t, \theta^{t-1}) = \text{argmax}_\theta E_{z \sim q}[\log p_\theta(\mathbf{X}, \mathbf{Z})]$. *** $\log p_\theta(\mathbf{X}) \geq M(q, \theta)$. $\forall q, \theta$; $\log p_\theta(\mathbf{X}) = M(q^*, \theta)$ when $q^* = p_\theta(\cdot | \mathbf{X})$, $\forall \theta$; EM algorithm does not guarantee a global optimum of $\log p_\theta(\mathbf{x})$, local optimum.**

3. Density Estimation

I. Maximum Likelihood Estimation

MLE $\hat{\theta}_n$ is the value of $\theta \in \Theta$ that maximizes the likelihood function $L_n(\theta) = \prod_{i=1}^n f(x_i; \theta)$. Properties: consistent $E[\hat{\theta}_n] = E[\theta]$; equivalent $\hat{\theta}_n$ is MLE of θ , then $g(\hat{\theta}_n)$ is MLE of $g(\theta)$; asymptotically normal $(\hat{\theta} - \theta^*)/\widehat{\text{se}} \sim N(0, 1)$; asymptotically optimal and efficient, the MLE has smallest variance with large samples; approximates the Bayes estimator.

II. Rao Cramer Bound

For any estimator $\hat{\theta}$ of θ it holds that $E_{x|\theta} \left[(\hat{\theta} - \theta)^2 \right] \geq \left(\frac{\partial_{\theta} b_{\theta} + 1}{E_{x|\theta}[\Lambda^2]} \right)^2 + b_{\theta}^2$, where $b_{\theta} = E_{x|\theta}[\hat{\theta}] - \theta$, $\Lambda = \frac{\partial}{\partial \theta} \log p(x|\theta) = \frac{1}{p(x|\theta)} \frac{\partial}{\partial \theta} p(x|\theta)$. *** Unbiased estimator $b_{\theta} = 0 \Rightarrow E_{x|\theta} \left[(\hat{\theta} - \theta)^2 \right] \geq \frac{1}{E_{x|\theta}[\Lambda^2]}$; tradeoff $\frac{\partial}{\partial \theta} b_{\theta} < 0$ vs. $b_{\theta} = 0$, unbiased estimator might not the best.**

III. Bayesian Inference

θ is a random variable with distribution $p(\theta|\chi)$; $X \sim p(\mathbf{x})$ where $p(\mathbf{x})$ unknown; $p(\mathbf{x}|\theta)$ is model reflect belief about x given θ . $p(\mathbf{x} = x|\chi) = \int p(\mathbf{x}, \theta|\chi) d\theta =$

$\int p(\mathbf{x}|\theta, \chi) p(\theta|\chi) d\theta = \int p(\mathbf{x}|\theta) p(\theta|\chi) d\theta \approx \int p(\mathbf{x}|\theta) \delta(\theta - \hat{\theta}) d\theta = p(\mathbf{x}|\hat{\theta})$.

4. Regression

I. Liner Regression

$y_i = \beta^T x_i + \epsilon_i$, where $\epsilon_i \sim N(0, \sigma^2)$. OLS minimizes $\hat{\beta} = \text{argmin}_{\beta} \frac{1}{n} \sum_{i=1}^n (y_i - \beta^T x_i)^2$. Matrix form $L(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T (\mathbf{Y} - \mathbf{X}\beta) = \mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{X}\beta + \beta^T \mathbf{X}^T \mathbf{X}\beta$. $\frac{\partial L(\beta)}{\partial \beta} = -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\beta = 0 \Rightarrow \hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$. L2 (ridge): $\hat{\beta} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{Y}$ reduces variance. *** OLSE is unbiased under iid $\rightarrow E[\hat{\beta}] = \beta$.**

II. Bias-Variance Tradeoff

$\text{MSE} = \left(E[\hat{f}(\mathbf{x})] - f(\mathbf{x}) \right)^2 + \text{Var}[\hat{f}(\mathbf{x})] + E[\epsilon^2]$, the first term is bias squared, second term is variance.

III. Transformations

We can apply a transformation $\phi(\mathbf{x}) = (1, x, x^2, \dots, x^k)$ to capture non-linear relationships $y_i = \beta^T \phi(x_i) + \epsilon_i$, $\hat{\beta} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{Y}$, where $\Phi \in R^{n \times m}$ with $\phi(x_i)^T$.

IV. Causality

If f is a counterfactually invariant predictor: **1)** for the anti-causal scenario, $f(\mathbf{X}) \perp W|Y$; **2)** for the causal scenario without selection (but possibly confounded) $f(\mathbf{X}) \perp W$; **3)** for the causal scenario without confoundedness (but possibly with selection), $Y \perp X|W, X_W^{\perp}$ and $f(\mathbf{X}) \perp W|Y$.

5. Gaussian Processes

I. Bavesian Linear Regression

$p(\mathbf{Y}|\mathbf{X}, \beta, \sigma) = N(\mathbf{Y}|\mathbf{X}^T \beta, \sigma^2) = \exp \left(-\frac{1}{2\sigma^2} (Y - \mathbf{X}^T \beta)^2 \right)$. Extend multiple linear regression by having a prior $p(\beta) = N(\beta|0, \Lambda^{-1}) = \exp \left(-\frac{1}{2} \beta^T \Lambda \beta \right)$. Given $X \in R^{n \times d}$, $\mathbf{y} \in R^n$, we have $p(\beta|\mathbf{X}, \mathbf{y}) = N(\mu_\beta, \Sigma_\beta) \Rightarrow \log p(\beta|\mathbf{X}, \mathbf{y}) = -\frac{1}{2} \sigma_\epsilon^2 (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) - \frac{1}{2} \beta^T \Lambda \beta = -\frac{1}{2} \beta^T (\sigma_\epsilon^2 \mathbf{X}^T \mathbf{X} + \Lambda) \beta + \beta^T (\sigma_\epsilon^2 \mathbf{X}^T \mathbf{y})$, $\mu_\beta = (\mathbf{X}^T \mathbf{X} + \sigma^2 \Lambda)^{-1} \mathbf{X}^T \mathbf{y}$ and $\Sigma_\beta = \sigma^2 (\mathbf{X}^T \mathbf{X} + \sigma^2 \Lambda)^{-1}$.

II. Gaussian Processes

Given $\mathbf{y} = \mathbf{X}\beta + \epsilon$, where $\epsilon \sim N(0, \sigma^2 I)$. We have $E_{\beta, \epsilon}[\mathbf{y}] = \mathbf{X}^T E[\beta] + E[\epsilon] = \mathbf{X}^T 0 + 0 = 0$, $\text{Cov}[\mathbf{y}] = E_{\beta, \epsilon}[(\mathbf{X}\beta + \epsilon)(\mathbf{X}\beta + \epsilon)^T] = \mathbf{X}E[\beta\beta^T]\mathbf{X}^T + E[\epsilon\epsilon^T]E[\epsilon]E[\epsilon]^T = \mathbf{X}E[\beta\beta^T]\mathbf{X}^T + E[\epsilon\epsilon^T] = \mathbf{X}\Lambda\mathbf{X}^T + \sigma^2 I$. We can rewrite it as $\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \sim N \left(\mathbf{y} \middle| 0, \begin{bmatrix} k_{1,1} + \sigma^2 & \dots & k_{1,n} \\ \vdots & \ddots & \vdots \\ k_{n,1} & \dots & k_{n,n} + \sigma^2 \end{bmatrix} \right)$, $k_{i,j} = k(x_i, x_j) = x_i^T \Lambda x_j$ as the kernel function.

III. Prediction using Gaussian Processes

Given the conditional Gaussian Distribution $\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} \sim N \left(\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right)$ with Σ_{ij} positive semi-definite and $\mathbf{a}_1, \mathbf{u}_1 \in R^e$, $\mathbf{a}_2, \mathbf{u}_2 \in R^f$, $\Sigma_{11} \in R^{e \times e}$, $\Sigma_{12} \in R^{e \times f}$, $\Sigma_{22} \in R^{f \times f}$, $\Sigma_{21} \in R^{f \times e}$.

Then the distribution is $p(\mathbf{a}_2|\mathbf{a}_1 = \mathbf{z}) = N(\mathbf{u}_2 + \Sigma_{21} \Sigma_{11}^{-1}(\mathbf{z} - \mathbf{u}_1), \Sigma_{22} - \Sigma_{21} \Sigma_{11}^{-1} \Sigma_{12})$.

Require: n observed data $\mathbf{X} = (x_1, \dots, x_n)^T \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$, kernel function k , noise variance σ^2 , new data point $x_{n+1} \in \mathbb{R}^d$

$K \leftarrow (k(x_i, x_j))_{1 \leq i, j \leq n}$ $\triangleright$ Compute kernel matrix
 $\mathbf{k} \leftarrow (k(x_{n+1}, x_i))_{1 \leq i \leq n}$ $\triangleright$ Similarity new data point and observed data
 $\mu_{y_{n+1}} \leftarrow \mathbf{k}^T (K + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$ $\triangleright$ Mean of pred. distr.
 $\sigma_{y_{n+1}}^2 \leftarrow k(x_{n+1}, x_{n+1}) - \mathbf{k}^T (K + \sigma^2 \mathbf{I})^{-1} \mathbf{k}$ $\triangleright$ Variance of pred. distr.
return $N(\mu_{y_{n+1}}, \sigma_{y_{n+1}}^2)$ $\triangleright$ Return pred. distr.

IV. Kernels

Symmetry $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ for $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$;

K_{AA} is positive semi-definite for $A \subseteq \mathcal{X}$;

stationary if $\tilde{k}(\mathbf{x} - \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')$; **isotropic**

if $\tilde{k}(\|\mathbf{x} - \mathbf{x}'\|_2) = k(\mathbf{x}, \mathbf{x}')$.

*** Isotropic implies stationary.**

$k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$; $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^n x_i y_i = \mathbf{y}^T \mathbf{x} = k(\mathbf{y}, \mathbf{x})$; $\mathbf{a}^T \mathbf{K} \mathbf{a} = \sum_{i=1}^n \sum_{j=1}^n a_i a_j k(\mathbf{x}_i^T, \mathbf{x}_j) = \sum_{i=1}^n \sum_{j=1}^n a_i a_j \mathbf{x}_i^T \mathbf{x}_j = \sum_{i=1}^n \sum_{j=1}^n a_i a_j (\mathbf{x}_i^T \sum_{j=1}^n a_j \mathbf{x}_j) = \|\sum_{i=1}^n a_i \mathbf{x}_i\|^2 \geq 0$. $k(\mathbf{x}, \mathbf{x}') = c k_1(\mathbf{x}, \mathbf{x}')$; $\phi(\mathbf{x}) = \sqrt{c} \phi_1(\mathbf{x})$, $\phi^T(\mathbf{x}) \phi(\mathbf{z}) = c \phi_1^T(\mathbf{x}) \phi_1(\mathbf{z}) = c k_1(\mathbf{x}, \mathbf{z})$, $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$; $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}) \ \phi_2(\mathbf{x})]^T$, $\phi_1^T(\mathbf{x}) \phi(\mathbf{z}) = \phi_1^T(\mathbf{x}) \phi_1(\mathbf{z}) + \phi_2^T(\mathbf{x}) \phi_2(\mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$, $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') \Rightarrow (\phi_1^T(\mathbf{x}) \phi_1(\mathbf{z})) (\phi_2^T(\mathbf{x}) \phi_2(\mathbf{z})) = (\sum_{i=1}^n f_i(\mathbf{x}) f_i(\mathbf{z})) (\sum_{j=1}^n g_j(\mathbf{x}) g_j(\mathbf{z})) = \sum_{i,j} f_i(\mathbf{x}) f_j(\mathbf{x}) (f_i(\mathbf{z}) g_j(\mathbf{z})) = \sum_{i,j} h_{ij}(\mathbf{x}) h_{ij}(\mathbf{z}) = \phi_3^T(\mathbf{x}) \phi_3(\mathbf{z})$. $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{y}) f'(\mathbf{x}')$; $\phi(\mathbf{x}) = f(\mathbf{x}) \phi_1(\mathbf{x})$, $\phi^T(\mathbf{x}) \phi(\mathbf{z}) = f(\mathbf{x}) \phi_1^T(\mathbf{x}) \phi_1(\mathbf{z}) f(\mathbf{z}) = f(\mathbf{x}) k(\mathbf{x}, \mathbf{z}) f(\mathbf{z})$. $k(\mathbf{x}, \mathbf{x}') = \exp(k(\mathbf{x}, \mathbf{x}')) = \sum_{m=1}^\infty k_1(\mathbf{x}, \mathbf{x}')^m / m!$ as $m \rightarrow \infty$. RBF kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2\lambda^2} \|\mathbf{x} - \mathbf{x}'\|_2^2) = \exp(-\frac{1}{2\lambda^2} \|\mathbf{x} - \mathbf{x}'\|_2^2) = \exp(-\frac{1}{2\lambda^2} \|\mathbf{x}\|_2^2) \exp(\frac{1}{2\lambda^2} \mathbf{x}^T \mathbf{x}') \exp(-\frac{1}{2\lambda^2} \|\mathbf{x}'\|_2^2)$. $\exp(-\frac{1}{2\lambda^2} \|\mathbf{x}\|_2^2) \exp(\frac{1}{2\lambda^2} \mathbf{x}^T \mathbf{x}') \exp(-\frac{1}{2\lambda^2} \|\mathbf{x}'\|_2^2)$ valid as $\phi(\mathbf{x}) = x$ similar to $\exp(k_1(\mathbf{x}, \mathbf{x}'))$ as a whole valid as $k = f k_1 f$.

6. Validation & Uncertainty Quantification

I. Model Validation

Cross-validation estimates model performance by splitting data, training on some parts, and testing on others.

Bootstrap estimates the distribution of a statistical parameter by repeatedly sampling with replacement from the original dataset and computing the parameter for each sample. $S = \frac{1}{n} \sum_{b \in S} S(\mathbf{Z}^b)$, $\sigma^2(S) = \frac{1}{n} \sum_{b \in S} S(\mathbf{Z}^b) - S$.

II. Linear Uncertainty Quantification

Suppose $X \in R^{n \times d}$, $\hat{\beta}|\mathbf{X} \sim N(\beta, \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1})$, the unbiased estimator of σ^2 is $\hat{\sigma}^2 = \frac{1}{n-d} \sum_{i \leq n} (X\hat{\beta} - Y)^2$, an approximate $1 - \alpha$ confidence interval for β_j is $\hat{\beta}_j \pm z_{\alpha/2} \widehat{\text{se}}(\hat{\beta}_j)$, where $z_{\alpha/2}$ is the cdf of the standard Gaussian and $\widehat{\text{se}}(\hat{\beta}_j)$ is the j -th diagonal element of $\sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}$. **P-value**, $H_0: \theta^* \in \Theta_0$, $H_1: \theta^* \in \Theta_1$, $\Theta_0, \Theta_1 \in \Theta$, given a sample $X_1, \dots, X_n \sim p(\cdot | \theta^*)$ and a statistic $t(X_1, \dots, X_n)$. We have $p = \inf\{\alpha: t(x_1, \dots, x_n) \geq c\}$, which equivalents to $p = \sup_{\theta \in \Theta_0} \Pr(t(X_1, \dots, X_n) \geq t(x_1, \dots, x_n) | \theta)$.

7. Bayesian Neural Networks

I. Stochastic Gradient Descent

Initialize weights $w^{(0)}$, define learning rate function $t \rightarrow \alpha_t$. for $t = 1, \dots, T$ do
 Pick a point (x, y) from the dataset $\mathcal{Z}$.
 Update weights: $w^{(t)} \leftarrow w^{(t-1)} - \alpha_t \nabla_{\mathbf{w}} L(y, f(\mathbf{x} | w^{(t-1)}))$.
SGD only offers a point estimate for the weights and biases of the neural network.

II. Bayesian Neural Networks

$p(w|D) = p(w) p(D|w)$, use variational inference to approximate the posterior, find a tractable parametric distribution $q(w|\theta) \sim N(\mu, \sigma^2 I)$. $\theta^* = \text{argmin}_{\theta} KL(q(w|\theta) | p(w|D)) \Rightarrow E_{w \sim q(\cdot)} [F(w, \theta)]$, where $F(w, \theta) = \log q(w|\theta) - \log p(w) - \log p(D|w)$. Initialize estimates for μ and σ , and learning rate function $t \rightarrow \alpha_t$ for $t = 1, 2, \dots$ do
 Sample $\epsilon \sim N(0, I)$.
 Compute the objective function $F(w, \theta)$.
 Update μ and σ as follows:

$$\mu \leftarrow \mu - \alpha_t \left(\frac{\partial}{\partial w} F(w, \theta) + \frac{\partial}{\partial \mu} F(w, \theta) \right)$$

$$\sigma \leftarrow \sigma - \alpha_t \left(\epsilon^T \frac{\partial}{\partial w} F(w, \theta) + \frac{\partial}{\partial \sigma} F(w, \theta) \right)$$

III. Active Learning

Given a target space $A \subseteq \mathcal{X}$ and a sample space $S \subseteq \mathcal{X}$. f maps $x \in \mathcal{X}$ to f_x , $y_x = f_x + \epsilon_x$. $D_{n-1} = \{(x_i, y_i)\}_{i \leq n} \subseteq S \times R$, want to find $x_n \in S$ having the most information about f in A . **ITL** $x_n = \text{argmax}_{x \in S} (f_{x_1} | y_{x_1} | D_{n-1}) = \text{argmax}_{x \in S} \frac{1}{2} \log \left(\frac{\text{var}[y_{x_n} | D_{n-1}]}{\text{var}[y_{x_n} | f_A, D_{n-1}]} \right)$, when $f \sim GP(\mu, k)$.

Safe Bayesian optimization fits a GP f on $\{x_i, y_i\}_{i \leq n}$. $[l_n^f(x), u_n^f(x)]$ is 95% confidence interval of $E[f(x)]$, pessimistic estimate $S_n = \{x: l_n^f(x) \geq 0\}$, optimistic estimate $\hat{S}_n = \{x: u_n^f(x) \geq 0\}$. Having target set $A_n = \{x \in \hat{S}_n: u_n^f(x) \geq \max_{x' \in S_n} l_n^f(x')\}$ perform ITL.

IV. Batch Active Learning

This method selects a batch of data points in each iteration, instead of one data point. NP-hard, use **ProbCover** to approximate it.

Require: $X \subseteq \mathcal{X}$, $b \in \mathbb{N}$
Ensure: $L \subseteq X$ of size b
1: $G = (X, E)$ where $E = \{(x, x') : \|x - x'\| \leq \delta\}$
2: $L \leftarrow \emptyset$
3: **for** $i = 1, 2, \dots, b$ **do**
4: $\hat{x} \leftarrow \text{argmax}_{x \in X} \{x' : (x, x') \in E, x' \in X\}$
5: $L \leftarrow L \cup \{\hat{x}\}$
6: $E \leftarrow E \setminus \{\{\hat{x}\} \times (B_\delta(\hat{x}) \cap X)\}$

8. Convex Optimization

Transform **constrained optimization** to **unconstrained optimization**: $\min_{w \in R^d} f(w)$, $g_i(w) = 0$, $h_j(w) \leq 0$, $\forall i \leq m, j \leq n \Rightarrow L(w, \alpha, \lambda) = f(w) + \sum_{i \leq m} \lambda_i g_i(w) + \sum_{j \leq n} \alpha_j h_j(w)$, $\alpha_j \geq 0$, λ_i and α_j are Lagrange multipliers, solve $\min_{w \in R^d} \max_{\alpha \geq 0} \max_{\lambda \geq 0} L(w, \lambda, \alpha)$.

Primal: $\theta_p(w) = \max_{\alpha \geq 0, \lambda} L(w, \lambda, \alpha) \Rightarrow p^* = \min_{w \in R^d} \max_{\alpha \geq 0, \lambda} L(w, \lambda, \alpha)$

Dual: $\theta_d(w) = \min_{w \in R^d} L(w, \lambda, \alpha) \Rightarrow d^* = \max_{\alpha \geq 0, \lambda} \min_{w \in R^d} L(w, \lambda, \alpha)$

w* optimal primal, λ^* , α^* optimal dual, **weak duality:** $\theta(\lambda^$

I. Hard Margin SVM

Primal: Given training data n in form (x_i, z_i) , $z_i \in \{1, -1\}$ and $\phi(x_i) = y_i$. SVM tends to find optimal hyperplane $w^T y + w_0 = 0$ having the maximum margin: $\min \frac{1}{2} \|w\|^2, z_i(w^T y + w_0) \geq 1, \forall i \leq n$. $L(w, w_0, \alpha) = \frac{1}{2} w^T w - \sum_{i \leq n} \alpha_i [z_i(w^T y + w_0) - 1], \forall i \leq n$. $* O(e^3)$ computational complexity.

Dual: $\max \sum_{i \leq n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j z_i z_j y_i^T y_j, \alpha_i \geq 0, \sum_i \alpha_i z_i = 0$. $\frac{\partial L}{\partial w} = w - \sum_{i \leq n} \alpha_i z_i y_i \Rightarrow w^* = \sum_{i \leq n} \alpha_i z_i y_i, \frac{\partial L}{\partial w_0} = \sum_{i \leq n} \alpha_i z_i \Rightarrow \sum_{i \leq n} \alpha_i z_i = 0$. $L(w^*, w_0^*; \alpha) = \frac{1}{2} \sum_{i \leq n} \sum_{j \leq n} \alpha_i \alpha_j z_i z_j y_i^T y_j - w_0 \sum_{i \leq n} \alpha_i z_i + \sum_{i \leq n} \alpha_i^* \cdot O(n^3)$.

II. Soft Margin SVM

ξ_i slack: $\xi_i = 0$ correct (outside margin); $0 < \xi_i < 1$ correct (in margin); $\xi_i > 1$ incorrect.

Primal: $\min \frac{1}{2} \|w\|^2 + C \sum_{i \leq n} \xi_i, z_i(w^T y + w_0) \geq 1 - \xi_i, \xi_i \geq 0$. **Dual:** $L(w, w_0, \xi; \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_{i \leq n} \xi_i - \sum_{i \leq n} \alpha_i [z_i(w^T y + w_0) - 1 + \xi_i] - \sum_{i \leq n} \beta_i \xi_i$. $\max_{\alpha, \beta} \frac{1}{2} \sum_{i \leq n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j z_i z_j y_i^T y_j, 0 \leq \alpha_i \leq C, \sum_{i \leq n} \alpha_i z_i = 0, \frac{\partial L}{\partial w} = w - \sum_{i \leq n} \alpha_i z_i y_i \Rightarrow w^* = \sum_{i \leq n} \alpha_i z_i y_i, \frac{\partial L}{\partial w_0} = \sum_{i \leq n} \alpha_i z_i \Rightarrow \sum_{i \leq n} \alpha_i z_i = 0, \frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i \Rightarrow C = \alpha_i + \beta_i (\forall i \leq n)$. $* C \rightarrow \infty$ leads to hard margin SVM.

III. Non-linear SVM (Kernel SVM)

Feature map $\phi(x) = (x, x')$. $\max_{\alpha} \frac{1}{2} \sum_{i \leq n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$, then $w^* T \phi(x) = \sum_{i=1}^n \alpha_i^* y_i \phi(x_i)^T \phi(x_i)$. $* K(x, x') = \phi^T(x) \phi(x')$.

IV. Multiclass SVM

For all class $z \in \{1, 2, \dots, M\}$ we introduce w_z and define the margin m such that $(w_{z_i}^T y_i + w_{z,0}) - \max_{z \neq z_i} (w_z^T y_i + w_{z,0}) \geq m, \forall y_i \in Y$.

10. Ensemble Methods

I. Bias and Variance of Ensemble Methods
Ensemble methods combine multiple classifiers to create a more accurate predictive model. Given estimators $f_1(x), \dots, f_B(x)$, $\hat{f}(x) = \frac{1}{B} \sum_{i=1}^B f_i(x)$; $\text{bias}[\hat{f}(x)] = E[\hat{f}(x)] - E[Y|x] = \frac{1}{B} \sum_{i=1}^B f_i(x) - E[Y|x] = \frac{1}{B} \sum_{i=1}^B \text{bias}[f_i(x)]$; $\text{Var}[\hat{f}(x)] = E[\hat{f}(x) - E[\hat{f}(x)]]^2 = E[\frac{1}{B} \sum_{i=1}^B (f_i(x) - E[f_i(x)])^2] = \frac{1}{B^2} \sum_{i=1}^B \text{Var}[f_i(x)] + \frac{1}{B^2} \sum_{i \neq j} \text{Cov}[f_i(x), f_j(x)] \approx \sigma^2/B$. Given classifiers $c_1, \dots, c_B$, combine to $\hat{c}(x) = \text{sign}(\sum_{i=1}^B \alpha_i c_i(x))$, where α_i is the weight. $* \text{Variance can be reduced if covariance is small, unbiased estimators remain unbiased after averaging.}$

II. Bagging

Bagging (bootstrap aggregation) combines classifiers trained on bootstrap samples. **Require:** $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$
Ensure: $\hat{c}_{BAG}$ with majority vote of $\{c_1, c_2, \dots, c_B\}$
1: for $b = 1$ to B do
2: $Z^{*b} \leftarrow b$ -th bootstrap sample from Z
3: Construct classifier c_b based on Z^{*b}
4: end for
5: return $\hat{c}_{BAG}(x) = \text{sgn}(\sum_{i=1}^B c_i(x)) = 0$

* Reduces covariance, variance, not much on bias.

III. Random Forests

Combines multiple decision trees, each tree is trained on a bootstrap sample; at each split within each tree, only selected subset m of p features to train, usually $m \approx \sqrt{p}$. **Require:** $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}, n_{\min}$
Ensure: Ensemble of trees $\{c_b(x)\}_{b=1}^B$
1: for $b = 1$ to B do
2: Draw bootstrap sample $Z^* = \{(x_1^*, y_1^*), \dots, (x_n^*, y_n^*)\}$ of size n from Z $\rightarrow$ generation of a random forest tree
3: repeat
4: Select m features at random from p features
5: Pick the best feature/split-point among the m selected features
6: Split the node into two daughter nodes
7: until node size $n_{\min}$ is reached
8: end for
9: return Ensemble of trees $\{c_b(x)\}_{b=1}^B$

III. Boosting (AdaBoost)

Require: $Z = (x_1, y_1), \dots, (x_n, y_n)$
Ensure: Classifier $\hat{c}_{ADA}(x)$
1: Initialize observation weights $w_i = 1/n$
2: for $b = 1$ to B do
3: Fit a classifier $c_b(x)$ to the training data using weights w_i
4: Compute $\epsilon_b = \frac{\sum_{i=1}^n w_i |c_b(x_i) - y_i|}{\sum_{i=1}^n w_i}$ $\triangleright$ error of c_b
5: Compute $\alpha_b \leftarrow \log \frac{1 - \epsilon_b}{\epsilon_b}$ $\triangleright$ final weight of c_b
6: Set $\forall i: w_i \leftarrow w_i \exp(\alpha_b I(c_b(x_i) \neq y_i))$
7: end for
8: return $\hat{c}_{ADA}(x) = \text{sgn}(\sum_{b=1}^B \alpha_b c_b(x))$

AdaBoost.M1 is equivalent to forward stagewise additive logistic modeling using exponential loss. $F(x) = \log \frac{P(y=1|x)}{P(y=-1|x)} = \frac{\exp(F(x))}{1 + \exp(F(x))}$; the objective of AdaBoost $J(F) = \log \frac{P(y=1|x)}{P(y=-1|x)}$ is minimized with respect to $F(x): J(F) = E[\exp(-yF(x))]$.

11. Deep Learning

I. Deep Neural Networks

$f(x; \theta) = \phi(W^{(L)} \phi(W^{(L-1)} \dots \phi(W^{(1)} x)))$, $\theta = [W^{(1)}, \dots, W^{(L)}]$, non-linear $\phi: R \rightarrow R$, also called activation functions. Backpropagation is used. Loss is $J(f; y) = \frac{1}{2} \sum_{i \leq n} (y_i - W^{(L)} \phi(W^{(L-1)} \dots \phi(W^{(1)} x)))^2$.

$(\nabla_{W^{(L)}})^T = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial W^{(L)}}; (\nabla_{W^{(L-1)}})^T = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial z^{(L-1)}} \cdot \frac{\partial z^{(L-1)}}{\partial W^{(L-1)}}; \dots; (\nabla_{W^{(1)}})^T = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial z^{(L-1)}} \cdot \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \cdot \dots \cdot \frac{\partial z^{(2)}}{\partial W^{(1)}}$.

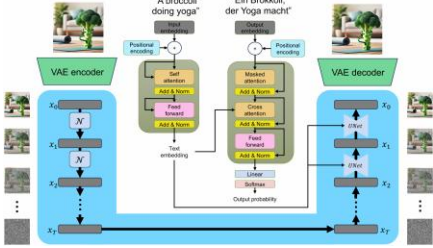
II. Variational Autoencoder (VAE)

InfoMax: $\arg \max_{\theta} I(X; Z) = \arg \max_{\theta} E[\log \frac{p(X,Z)}{p(X)p(Z)}] = \arg \max_{\theta} E[\log \frac{p(X|Z)p(Z)}{p(X)p(Z)}] = \arg \max_{\theta} E[\log p(X|Z)] + E[\log p(Z)] \approx \arg \max_{\theta} E[\log p(X|Z)]$.

* Although informative, but not about disentanglement or robustness.

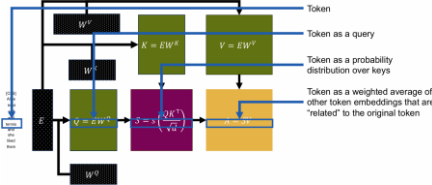
Autoencoder: $\arg \max_{\theta, p_\theta} \sum_{i \leq n} \log p_{\theta'}(x_i) \Rightarrow \log p_{\theta'}(x_i) = E[\log p_{\theta'}(x_i)] = E[\log \frac{p_{\theta'}(x_i|Z)p_\theta(Z)}{p_\theta(Z)}] = E[\log p_\theta(x_i|Z)] - E[\log \frac{q_\theta(Z|x_i)}{p_\theta(Z)}] = E[\log p_\theta(x_i|z)] - E[\log \frac{q_\theta(z|x_i)}{p_\theta(z)}] + E[\log \frac{q_\theta(z|x_i)}{p_\theta(z)}] = E[\log p_\theta(x_i|z)] - KL(q_\theta(z|x_i) || p_\theta(z)) + KL(q_\theta(z|x_i) || p_\theta(z|x_i)) \geq E[\log p_\theta(x_i|z)] - KL(q_\theta(z|x_i) || p_\theta(z)) = ELBO$

III. Transformer

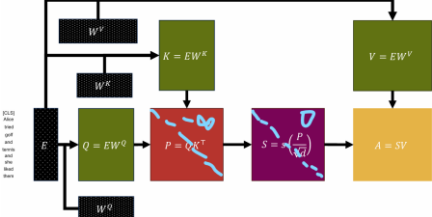


WordPiece Tokenization: 1) set $i = 0$; 2) while $i < S$ (S is predefined number of tokens), a) choose a subsequence length $l \in N$, b) find the most common subsequence s of length l over the training corpus, c) let i -th token represent s , d) $i = i + 1$.

Self-attention (cross attention – another encoding):

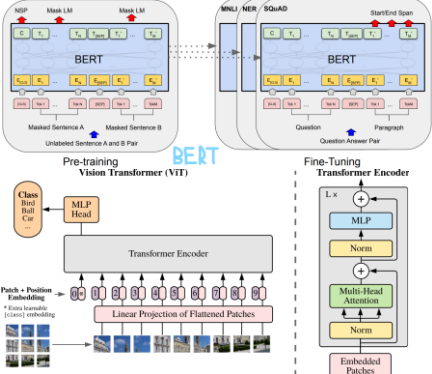


Masked self-attention: only looks backwards



Positional encoding: the positional encoding matrix p has T rows and d columns. T is the maximum number of tokens in a sentence. d is the dimensionality of the positional embedding. The encoding i is defined as $p_{i,2j} = \sin(\alpha^{-2j/i})$ and $p_{i,2j+1} = \cos(\alpha^{-2j/i})$ for $j \leq d/2$ and $\alpha = 10^4/d$.

Other transformers: BERT and vision



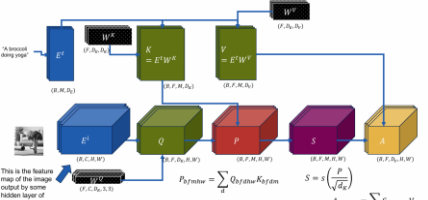
IV. Stable Diffusion

Master equation $\tau \frac{d}{dt} c_i = \frac{p-q}{2} (c_{i+1} - c_{i-1}) + \frac{p+q}{2} (c_{i+1} - 2c_i + c_{i-1})$. **Diffusion result** $\frac{\partial}{\partial t} c(x, t) = D \frac{\partial^2}{\partial x^2} c(x, t) \Rightarrow \hat{c}(R, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{iRx} c(x, t) dx$. **Inverse Fourier transformation** $c(x, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-iRx} \hat{c}(R, t) dR = \frac{1}{\sqrt{4\pi Dt}} \exp(-\frac{(x-x_0)^2}{4Dt})$.

V. Convolution & Deconvolution



VI. Multiheaded Cross-attention with U-Net



12. Non-parametric Bayesian Methods

I. EM Optimization

Let $\chi = \{x_1, \dots, x_n\}$ be the observable variables $Z = \{z_1, \dots, z_m\}$ be the latent variables. The EM method is to repeat the following procedure until the convergence of θ . **1) Expectation:** calculate a function Q of θ given previous estimated $\theta^{(j)}$, $Q(\theta; \theta^{(j)}) = E[L(\chi, Z|\theta) | \chi, \theta^{(j)}]$, where $L(\chi, Z|\theta) = \log p(\chi, \chi_l | \theta)$ log-likelihood function. **2) Maximization:** estimate new parameter $\theta^{(j+1)}$ by maximizing the function Q , $\theta^{(j+1)} = \arg \max_{\theta} Q(\theta; \theta^{(j)})$. $* \text{Guaranteed to converge to a point with gradient of 0.}$

II. K-Means

Given $\chi = \{x_1, \dots, x_n\}$. K-means finds an assignment function $c(x): R^d \rightarrow \{1, \dots, k\}$ and prototypes $Y = \{\mu_1, \dots, \mu_k\} \subset R^d$ that are minimizers of $c(x) \in \arg \min_c \sum_x \|x - \mu_{c(x)}\|_2^2$. while $c(x)$ and μ_k keep changing do

$c(x) = \arg \min_{k \in \{1, \dots, k\}} \|x - \mu_k\|_2^2$ $\triangleright$ E-step: assign x to the nearest prototypes
 $\mu_j = \frac{1}{|\{x: c(x) = j\}|} \sum_{x: c(x)=j} x$ $\triangleright$ M-step: update prototypes
return $c(x)$, $\forall x \in \chi$

III. Mixture Models

while $c(x)$ and μ_k keep changing do
 $\gamma_{kc} = \frac{p(x | c, \theta^{(j)}) p(c | \theta^{(j)})}{p(x | \theta^{(j)})}$ $\triangleright$ E-step: soft-assign sample x to clusters
 $\mu_k^{(j+1)} = \frac{\sum_x \gamma_{kc} x}{\sum_x \gamma_{kc}}$, $\Sigma_k^{(j+1)} = \frac{\sum_x \gamma_{kc} (x - \mu_k^{(j+1)})^2}{\sum_x \gamma_{kc}}$ $\triangleright$ M-step: update parameters
return $c(x)$, $\forall x \in \chi$

IV. Dirichlet Process

Dirichlet distribution is the multivariate generalization of the beta distribution
 $\text{Beta}(\theta; a, b) = \frac{1}{\beta(a,b)} \theta^{a-1} (1 - \theta)^{b-1}$, where $\beta(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$, where $\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$. Given $x = \{x_1, \dots, x_n\}$ ($x_i \in [0, 1]$) $\alpha = \{\alpha_1, \dots, \alpha_n\}$ ($\alpha_i > 0$). $\text{Dir}(x, \alpha) = \frac{1}{\beta(\alpha)} \prod_{k \leq n} x_k^{\alpha_k - 1}$, where $\beta(\alpha) = \frac{\prod_{k \leq n} \Gamma(\alpha_k)}{\Gamma(\sum_{k \leq n} \alpha_k)}$. **Expected number of clusters in DP mixture:** $x_1 \sim F_0, x_i | x_1 \dots x_{i-1} = X \sim F_{i-1}$ (if $p = \frac{i-1}{\alpha+k-1}$), $X \sim F_0$ (if $p = \frac{\alpha}{\alpha+k-1}$). $S(n) = E[\sum_{k \leq n} 1\{x_k \in \{x_1, \dots, x_{k-1}\} = \emptyset\}] = \sum_{k \leq n} P(x_k \in \{x_1, \dots, x_{k-1}\} = \emptyset) = \sum_{k \leq n} P(x_k \sim F_0) = \sum_{k \leq n} \frac{1}{\alpha+k-1} \cdot S(n) - 1 + \frac{\alpha}{\alpha+n} \leq I(n) \leq S(n)$, $I(n) \leq S(n) \leq I(n) + 1 - \frac{\alpha}{\alpha+n}$, $I(n) = \int_1^{n+1} \frac{\alpha}{\alpha+x-1} dx = \alpha \int_1^{n+1} \frac{1}{x} dx = \alpha(\ln(\alpha+n) - \ln(\alpha))$, as $n \rightarrow \infty$, $S(n) \sim \alpha \ln(n)$, $S(n) \in \Theta(\alpha \ln(n))$.

V. Chinese Restaurant Process

$p(\text{customer } n+1 \text{ joins table } \tau | P) = \frac{|\tau|}{\alpha+n+1}, \tau \in P; p = \frac{\alpha}{\alpha+n}, \tau \notin P$. It is exchangeable $p(P) = \frac{\alpha!}{|\text{pos}(S(n-1)+\alpha+1)} \prod_{1 \leq i \leq |P|} (|\tau_i| - 1)!$. **DPMM:** $p(z_i = k | z_{-i}, x, \alpha, p) = p(z_i = k | z_{-i}, \alpha) p(x_i | z_{-i}, x_i = k, \mu)$. $p(x_i | \mu, z_i = k, z_{-i}, x_{-i}) = p(x_i | x_{-i}, \mu)$ (k exists), $p(x_i | \mu)$.

13. Probably Approximately Correct (PAC)

I. Empirical Risk Minimization

Generalization: $R(\hat{c}) = P(\hat{c}(X) \neq c(X))$ measures how well the hypothesis $\hat{c}$ performs on the entire distribution. **Empirical:** $\hat{R}_n(\hat{c}) = \frac{1}{n} \sum_{i \leq n} 1\{\hat{c}(x_i) \neq y_i\}$ average error of hypothesis $\hat{c}$ on the given sample of size n . Minimization is to select the classifier $\hat{c}_n^* \in C$ with smallest training error $\hat{c}_n^* = \arg \min_c \hat{R}_n(c)$, where $\hat{R}_n(c) = \frac{1}{n} \sum_{i \leq n} 1\{c(x_i) \neq y_i\}$.

II. PAC Learning Model

A learning algorithm A can learn a concept class C from H , if there is a polynomial function such that 1) for any concept $c \in C$ 2) for any distribution D on $\chi\{0,1\}$ and 3) for any $0 < \epsilon < 1/2$, $0 < \delta < 1/2$, $P_{Z \sim D^n}(R(\hat{c}) \leq \epsilon) \geq 1 - \delta$, $n \geq \text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, \text{size}(c))$. $* A$ runs in polynomial time in $1/\epsilon$ and $1/\delta$ is efficient.

II. General PAC Learning Model

A learning algorithm A can learn a concept class C from H , if there is a polynomial function such that D on $\chi\{0,1\}$ and $0 < \epsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$, $P_{Z \sim D^n}(R(\hat{c}) - \inf_{c \in C} R(c) \leq \epsilon) \geq 1 - \delta$, where $n \geq \text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, \text{size}(c))$. **Error bound finite C :** $P(R(\hat{c}_n^*) - \inf_{c \in C} R(c) > \epsilon) \leq 2|C| \exp(-2n\epsilon^2)$; **Error bound infinite C :** $P(R(\hat{c}_n^*) - \inf_{c \in C} R(c) > \epsilon) \leq 9n^{VC_C} \exp(-\frac{ne^2}{32})$

III. Vapnik-Chervonenkis Inequality

$R(\hat{c}_n^*) - \inf_{c \in C} R(c) = R(\hat{c}_n^*) - R(\hat{c}_n^*) + R(\hat{c}_n^*) - \inf_{c \in C} R(c) \leq R(\hat{c}_n^*) - R(\hat{c}_n^*) + R(\hat{c}_n^*) - R_n(c^*) \leq \sup_{c \in C} |R(\hat{c}_n^*) - R_n(c^*)| + \sup_{c \in C} |R(\hat{c}_n^*) - R(c)|$, where $c^* = \arg \min_c R(c)$. The bound requires uniform convergence since $\hat{c}_n^*$ is minimized with the empirical risk $P(R(\hat{c}_n^*) - \inf_{c \in C} R(c) > \epsilon) \leq P(\sup_{c \in C} |R(\hat{c}_n^*) - R(c)| > \frac{\epsilon}{2})$, ERM works if $\sup_{c \in C} |R(\hat{c}_n^*) - R(c)| \rightarrow 0$. **VC**

dimension is a measure of the capacity of a set of functions that can be learned by a **binary** classification algorithm.

Classifier	VC-dim
stump classifier	$(-\infty, a], a \in \mathbb{R}$ 1
all intervals in $\mathbb{R}$	$\{[a, b] a, b \in \mathbb{R}\}$ 2
all unions of k intervals in $\mathbb{R}$	$\{\cup_{i=1}^k [a_i, b_i] a_i, b_i \in \mathbb{R}\}$ 2k
all half-planes in $\mathbb{R}^2$	$\{(x, y) ax + by + c \geq 0, a, b, c \in \mathbb{R}\}$ 3
all convex polygons in $\mathbb{R}^2$	- ∞
all convex polygons with at most k - vertices in $\mathbb{R}^2$	$2k + 1$

Error bound (finite C bounded by N):

$P(\sup_{c \in C} |R(\hat{c}_n^*) - R(c)| > \epsilon) = U_{c \in C} P(|R(\hat{c}_n^*) - R(c)| > \epsilon) \leq \sum_{c \in C} P(|R(\hat{c}_n^*) - R(c)| > \epsilon) \leq 2N \exp(-2n\epsilon^2)$. $R(c) \leq \hat{R}_n(c) + \sqrt{\frac{\ln N - \ln(\delta/2)}{2n}}$, where $\epsilon = \sqrt{\frac{\ln N - \ln(\delta/2)}{2n}}$, $\delta = 2N \exp(-2n\epsilon^2)$.

IV. Learning Hyperplanes with ER

Assume X has a density. If $\hat{c}$ is the ERM classifier, then for all possible distributions of (X, Y) , if $n \geq d$ and $2 \frac{d}{n} \leq \epsilon \leq 1$, we have $P(R(\hat{c}) - \inf_{c \in C} R(c) > \epsilon) \leq e^{2de} (2 \binom{n}{d} + 1) e^{-\frac{n\epsilon^2}{2}}$.

Moreover, if $n \geq d$, $E[R(\hat{c}) - R] \leq \sqrt{\frac{2}{n}} (d + 1) (\ln n + 2)$. $* (\frac{n}{d})$ grows polynomial for d , so learnability is slow.