Statistical Machine Learning Models

Max Chen

Abstract

This is the summary notes by Yuling Max Chen, covering the key concepts and frequently-used models in statistical machine learning. The structure and some proofs are referred to the lecture note of *Statistical Machine Learning* by Prof François Caron, as well as the lecture notes of *Advanced Topics in Statistical Machine Learning* by Prof Tom Rainforth. For detailed explanations and/or examples, please read the original notes. Some personal ideas are also added, hence may not be 100% theoretically rigorous but should be helpful for the comprehension of the materials.

No person or party should use this notes for any purpose other than studying and understanding the notes itself.

Contents

1 Unsupervised Learning

Autoencoder: Let $h_\theta : \mathbb{R}^p \to \mathbb{R}^p$ be the function defined by

$$h_\theta(x_i) = \text{dec}_\theta(\text{enc}_\theta(x_i)) = \widehat{x}_i$$

where $z_i = enc_\theta(x_i)$ is the latent representation.
Empirical Risk Minimization: For risk of autoencoder h_θ and loss function L $R(h_\theta) = \mathbb{E}[L(X, h_\theta(X))]$, the empirical risk is $\widehat{R}_n(h_\theta) = \frac{1}{n}\sum_{i=1}^n L(x_i, h_\theta(x_i))$.

1.1 Principle Component Analysis (PCA)

A dimension reduction tool that aims to find a new basis to represent a noisy data, often used for data preprocessing.

PCA finds an orthogonal basis (principal components) $v_1, \cdots, v_p$ such that:
• 1st PC (v_1) is the direction of the highest variance;
• ith PC (v_i) is the direction orthogonal to $v_1, \cdots, v_{i-1}$ of the highest variance.

K-dim representation of the data: $z_i = V_{1:K}^\top x_i = \left(v_1^\top x_i, \ldots, v_K^\top x_i\right)^\top \in \mathbb{R}^K$, where $V_{1:K}$ is $p \times K$
Reconstructed x_i: $\hat{x}_1 = V_{1:K} z_i = V_{1:K} V_{1:K}^T x_i$, where $V_{1:K} \in \underset{A_{p\times K}\in\mathcal{A}}{\arg\min} \frac{1}{n}\sum_{i=1}^n \|x_i - AA^T x_i\|^2$

PCA as ERM: Let $h_A(x_i) = AA^T x_i$ be the autoencoder parameterised by an orthonormal $p \times K$ matrix A, then the risk autoencoder for a squared loss is $R(h_A) = \mathbb{E}[\|X - AA^T X\|^2]$ and the empirical risk is $\hat{R}_n(h_A) = \frac{1}{n}\sum_{i=1}^n \|x_i - AA^T x_i\|^2$, and:
•$V_{1:K}^* = (v_1^*, \cdots, v_k^*) = \arg\min R(h_A)$ and $V_{1:K} = (v_1, \cdots, v_k) = \arg\min \hat{R}_n(h_A)$

Eigen-Decomposition of the Covariance Matrix: $\Sigma = V^*\Lambda^* V^{*T}$, where:
•Σ is a *real* and *symmetric* $p \times p$ matrix ($\exists(v_1^*, \cdots, v_p^*)$ and $\lambda_1^* \geq \cdots, \geq \lambda_p^*$ s.t. $\Sigma v_i^* = \lambda_i^* v_i^*$), and *positive semi-definite* ($\lambda_i^* \geq 0$);
• $V^* = (v_1^*, \cdots, v_p^*)$ and $\Lambda^* = (\lambda_1^*, \cdots, \lambda_p^*)$.

Total Variance of the random vector X:

$$\begin{aligned}\text{TV}(X) &= \mathbb{E}\left[\sum_{j=1}^p (X_j - \mathbb{E}[X_j])^2\right] = \sum_{j=1}^p \Sigma_{jj} = \text{trace}(\Sigma) \\ &= \text{trace}\left(V^*\Lambda^*(V^*)^\top\right) = \text{trace}\left((V^*)^\top V^*\Lambda^*\right) = \text{trace}(\Lambda^*) = \sum_{j=1}^p \lambda_j^*\end{aligned}$$

Sample Covariance Matrix (S): supposed X centered,

$$S = \frac{1}{n-1}\sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^\top = \frac{1}{n-1}\sum_{i=1}^n x_i x_i^\top = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$$

• S is *real, symmetric, positive semi-definite.*
• **Eigen-decomp of S**: $S = V\Lambda V^T$, with $\lambda_1 \geq, \cdots, \geq \lambda_p$ and $TV(X) = \sum_{i=1}^p \lambda_i$.

Derivation of PCs:
• **First PC**: $Z_1 = a_1^T X = \sum_{j=1}^p a_{1j} X_j \implies Var(Z_1) = a_1^T \Sigma a_1$
$\implies v_1^* = \underset{a_1 \in \mathbb{R}^1}{\arg\max} a_1^T \Sigma a_1$ subject to $a_1^T a_1 = 1$
$\implies \mathcal{L}(a_1, \gamma_1) = a_1^T \Sigma a_1 - \gamma_1 (a_1^T a_1 - 1) \implies \frac{\partial \mathcal{L}(a_1, \gamma_1)}{\partial a_1} = 2\Sigma a_1 - 2\gamma_1 a_1 = 0 \implies \Sigma a_1 = \gamma_1 a_1$
$\implies$ The 1st PC is the eigenvector v_1^* associated with the largest eigenvector λ_1^* and $Var(z_1) = a_1^T \Sigma a_1 = \gamma_1^*$
• **Second and subsequent PCs**:
$v_2^* = \underset{a_2 \in \mathbb{R}^p}{\arg\max} a_2^\top \Sigma a_2$ subject to: $a_2^\top a_2 = 1$ and $a_2^\top v_1^* = 0$
$\implies \mathcal{L}(a_2, \gamma_2, \mu) = a_2^\top \Sigma a_2 - \gamma_2 \left(a_2^\top a_2 - 1\right) - \mu \left(v_1^*\right)^\top a_2 \implies \frac{\partial \mathcal{L}(a_2, \gamma_2, \mu)}{\partial a_2} = 2\Sigma a_2 - 2\gamma_2 a_2 - \mu v_1^* (*)$
$\overset{a_2^T \times (*)}{\implies} a_2^T \Sigma a_2 = \gamma_2 \implies \Sigma a_2 = \gamma_2 a_2 \implies a_2$ is the e-vector of Σ associated with the 2nd largest eigenvalue λ_2
$\overset{v_1^{*T} \times (*)}{\implies} 2 v_1^{*T} \Sigma a_2 = \mu = 2 a_2^T \Sigma v_1^{*T} = \lambda_1^* a_2^T v_1^{*T} = 0$
• **Empirical PCs**: replace Σ with S.

Properties of PCs:
• Projection onto the principal components have variance/sample-variance given by the eigenvalues of Σ/S: $\text{var}(Z_j) = \text{var}\left(\left(v_j^*\right)^\top X\right) = \lambda_j^*$ and $\frac{1}{n-1}\sum_{i=1}^n z_{ij}^2 = \lambda_j$
• Projections onto the principal components are uncorrelated, for $j \neq k$:
$\text{cov}(Z_j, Z_k) = \left(v_j^*\right)^\top \Sigma v_k^* = \lambda_k^* \left(v_j^*\right)^\top v_k^* = 0$ and $\frac{1}{n-1}\sum_{i=1}^n z_{ij} z_{ik} = v_j^\top S v_k = \lambda_k v_j^\top v_k = 0$
• Proportion of total variance explained by the first K (population) PCs: $\frac{\sum_{j=1}^K \lambda_j^*}{\sum_{j=1}^p \lambda_j^*}$
Comments on PCA:
(i) Assuming large variance is meaningful;
(ii) The PCs depend heavily on the units measurement. Where the data matrix contains measurements of vastly differing orders of magnitude, the PC will be greatly biased in the direction of larger measurement.
• Either calculate PC using corr(X) instead of cov(X), or standardize the data before doing the PCs.
(iii) Lack of robustness to outliers: the variance is affected by outliers and so are PCs.
(iv) (Sub-)Sample sizes have an effect on PCs.
(v) If $n < p$, then S is singular (some eigenvalues are 0), hence cannot get p PCs. But PCA is still possible (since PCA does not involve matrix inverse), as long as we are not interesting in all the p PCs.

Singular Value Decomposition (SVD): Any real-valued $n \times p$ matrix $\mathbf{X}$ can be written as $X = UDV^T$, where:
(i) U, V are 2 $n \times n$ orthogonal matrices;
(ii) D is a $n \times p$ matrix with decreasing non-negative elements on the diagonal (the singular values) and zero off-diagonal elements.
• Let X be $n \times p$ data matrix, then the sample covariance matrix is:

$$S = \frac{1}{(n-1)}\mathbf{X}^\top \mathbf{X} = \frac{1}{(n-1)}\left(UDV^\top\right)^\top \left(UDV^\top\right) = \frac{1}{(n-1)} V D^\top U^\top U D V^\top = V \frac{1}{(n-1)} D^\top D V^\top = V \Lambda V^T$$

• The **Gram Matrix**: $\mathbf{B} = \mathbf{X}\mathbf{X}^\top = \left(UDV^\top\right)\left(UDV^\top\right)^\top = UDV^\top V D^\top U^\top = UDD^\top U^\top =$

$(UDV^TV)(VV^TD^TU^T) = (\mathbf{X}V)(\mathbf{X}V)^T = \mathbf{Z}\mathbf{Z}^T$, i.e. the decomp of the Gram matrix gives the transformed variable with less computation than calculating the eigen-decomp of S if $p > n$.

1.2 k-Means

Partition: $\Pi = \{C_1, \cdots, C_K\}$ of the set of integers $\{1, \cdots, n\}$ is s.t. $\forall k, k'(k \neq k') \in \{1, \cdots, K\}$: (i) $C_k \neq$ and $C_k \subseteq \{1, \cdots, n\}$; (ii) $C_k \cap C_{k'} =$; (iii) $\bigcup_{k=1}^{K} C_k = \{1, \cdots, n\}$

Model-free Clustering: A map $\mathcal{F} : (d, \rho) \to \{C_1, \cdots, C_k\}$, where $d = (x_1, \cdots, x_n) \in \mathbb{R}^{p \times n}$ be the dataset and $\rho : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}_+$ be a dissimilarity measurement, e.g. $\rho(x_i, x_j) = \|x_i - x_j\|^2$. There are 3 properties:
(i) Scale invariance: $\forall \alpha > 0, \mathcal{F}(d, \alpha\rho) = \mathcal{F}(d, \rho)$
(ii) Richness: $\forall \Pi = \{C_1, \ldots, C_K\}$ of $\{1, \ldots, n\}$, $\exists \rho : \mathcal{F}(d, \rho) = \Pi$
(iii) Consistency: If ρ and ρ' are two dissimilarities such that $\forall x_i, x_j$ the following holds:

$$x_i, x_j \text{belong to the same cluster in} \mathcal{F}(d, \rho) \Longrightarrow \rho'(x_i, x_j) \leq \rho(x_i, x_j)$$
$$x_i, x_j \text{belong to different clusters in} \mathcal{F}(d, \rho) \Longrightarrow \rho'(x_i, x_j) \geq \rho(x_i, x_j)$$

then $\mathcal{F}(d, \rho') = \mathcal{F}(d, \rho)$.
(Kleinberg (2003) proved that there exists no clustering method that satisfies all three properties. Every algorithm therefore has to find a trade-off between the above three properties.)

Extended K-means Objective Function: The dissimilarity of a cluster C_k will be expressed as the sum of the dissimilarities of pairs of data in that cluster, divided by the size of the cluster:

$$\widetilde{W}(C_1, \ldots, C_K) = \frac{1}{2}\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{i,i' \in C_k}\rho(x_i, x_{i'}) = \frac{1}{2}\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{i,i' \in C_k}\|x_i - x_{i'}\|^2$$
$$= \min_{\mu_1,\ldots,\mu_K}\sum_{k=1}^{K}\sum_{i \in C_k}\|x_i - \mu_k\|^2 = \min_{\mu_1,\ldots,\mu_K}\sum_{i=1}^{K}\|x_i - \mu_{z_i}\|^2, \text{ where } z_i = k \iff i \in C_k$$

• Because $\frac{1}{2|C_k|}\sum_{i,i' \in C_k}\|x_i - x_{i'}\|^2 = \sum_{i \in C_k}\|x_i - \bar{x}_{C_k}\|^2$ proof?
And $\bar{x}_{C_k} = \underset{\mu_k \in \mathbb{R}^p}{\arg\min}\sum_{i \in C_k}\|x_i - \mu_k\|^2$.

Then, to minimize $\tilde{W}(C_1, \cdots, C_k)$, cannot simultaneously minimize $(C_1, \cdots, C_K)$ and $(\mu_1, \cdots, \mu_K)$, but can instead minimize via coordinate descent method, i.e. optimize over one set of the parameters while keeping the rest fixed.

Algorithm 1: K-means Clustering

Randomly initialize K clustering centroids $\mu_1, \cdots, \mu_K$
while *centroids not converge* **do**
  for $i = 1, ...n$ **do**
    Clustering assignment: $z_i := \arg\min_k \|x_1 - \mu_k\|^2$
  end
  for $k = 1, ..., K$ **do**
    Set $C_k = \{i : z_i = k\}$ for each k.
    Move centroids: $\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$
  end
end
Return the partition $\{C_1, ..., C_k\}$ and the centroids $\{\mu_1, ..., \mu_k\}$.

K-means as ERM:
For parameter $\theta = (\mu_1, \cdots, \mu_K)$,

$$z_i = \text{enc}_\theta(x_i) = \arg\min_{k=1,...,K} \|x_i - \mu_k\|^2$$
$$\widehat{x}_i = \text{dec}_\theta(z_i) = \mu_{z_i}$$

Then the risk is defined as $R(h_\theta) = \mathbb{E}[L(X, h_\theta(X))] = \mathbb{E}\left[\|X - h_\theta(X)\|^2\right]$, where $h_\theta(x) = dec_\theta(enc_\theta(x_i))$. Hence the empirical risk is:

$$\begin{aligned}\widehat{R}_n(h_\theta) &= \frac{1}{n}\sum_{i=1}^n \|x_i - h_\theta(x_i)\|^2 \\ &= \min_{z_1,...,z_n} \frac{1}{n}\sum_{i=1}^n \|x_i - \mu_{z_i}\|^2 \\ &= \min_{C_1,...,C_K} \frac{1}{n} W(C_1, \ldots, C_K, \mu_1, \ldots, \mu_K)\end{aligned}$$

And $\hat{\theta} = \arg\min_\theta \hat{R}_n(h_\theta)$.

Choice of K: The K-mean objective funtion $\tilde{W}$ will always decreases for large K, but larger K does not mean good. The **Calinski-Harabasz score** measures the separability of the clusters: higher value means well-separated and dense cluster.

$$CH = \frac{\sum_{k=1}^K |C_k| \times \|\mu_k - \bar{x}\|^2}{\sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2} \times \frac{n - K}{K - 1}$$

Stochastic Optimization for K-means: For large dataset, the coordinate descent method of K-means clustering is time consuming, as each time the centroids are updated after a loop of clustering assignment over all the data points. We instead update the centroids and clustering assignment iteratively for each randomly chosen data point.
Vector Quantisation: (for loosy data compression)
A data matrix $\mathbf{X}$ contains $n \times p$ real numbers, vector quantisation can restore $\mathbf{X}$ with 2 components:
• the codebook of K codewords: $\theta = (\mu_1, \cdots, \mu_K)$ ($K \times p$ real numbers) and;
• the clustering assignment $(z_i)_{i=1}^n$ for each data point ($\lceil \log K \rceil \times n$ bits).

Algorithm 2: Stochastic Optimization for K-means

Initialize step size (α_t) such that $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$.
Initialize randomly K clustering centroids $(\mu_1, \cdots, \mu_K)$.
Initialize $t = 1$.
while *not converge* **do**
  Randomly pick x_i. (Or pick in order)
  Clustering assignment: $z_i := \arg\min_{k=1,\ldots,K} \|x_i - \mu_k\|^2$
  Update clustering centroids: $\mu_{z_i} := \mu_{z_i} + \alpha_t(x_i - \mu_{z_i})$
  $t = t + 1$
end

DP-means Clustering:

Algorithm 3: DP-means Clustering

Initialize K = 1 and $\mu_1 = \frac{1}{n}\sum x_i$ (global mean).
Initialize $\lambda > 0$.
while *not converge* **do**
  DP-means cluster assignment:
  for $i = 1, ..., n$ **do**
    if $\min_{k=1,\ldots,K} \|x_i - \mu_k\|^2 > \lambda$ **then**
      K = K + 1
      $c_i = K$
      $\mu_K = x_i$
    end
    else
      $c_i = \arg\min_{k=1,\ldots,K} \|x_i - \mu_k\|^2$
      Update the previous and the current cluster centroid that data point i belongs to.
    end
  end
end

2 Supervised Learning

Supervised Learning requires labeled data. And it can be categorized as Discriminant Analysis and Generative Analysis.

2.1 The Basics

Prediction rule: $h : \mathcal{X} \to \mathcal{Y}$.
Classifier: $h(x) = \arg\max_{k=1,\ldots,K} f_k(x)$, where $f_k(x) : \mathcal{X} \to \mathbb{R}$ are **discriminant functions**.
Decision Boundary: For binary classifier, $f(x) = 0$; For multi-class classifier, $f_k(x) = f_{k'}(x)$
Linear Prediction Rule: $\begin{cases} h(x) = \beta_0 + \beta^\top x, \text{for regression} \\ f_k(x) = \beta_{k0} + \beta_k^\top \text{for classification} x \end{cases}$.

• **Least square linear regression**: $\widehat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x}) y_i}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad \widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1 \bar{x}$

• **Binary Classifier**: $\widehat{h}^{(d)}(x) = \begin{cases} 1 & \text{if } \widehat{f}^{(d)}(x) \geq 0 \\ -1 & \text{otherwise} \end{cases} \implies \widehat{f}^{(d)}(x) = \sum_{i=1}^n 1_{\lfloor x_i \rfloor = \lfloor x \rfloor} \left(1_{y_i=1} - 1_{y_i=-1}\right)$

Decomposition of the variance:

$$\begin{aligned} \operatorname{cov}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top\right] &= \underbrace{\mathbb{E}[\operatorname{cov}(X \mid Y)]}_{\text{within-class covariance}} + \underbrace{\operatorname{cov}(\mathbb{E}[X \mid Y])}_{\text{between-class covariance}} \\ &= \sum_{k=1}^K \pi_k \Sigma_k + \sum_{k=1}^K \pi_k \left(\mu_k - \mu\right)\left(\mu_k - \mu\right)^\top \end{aligned}$$

Separability: $\frac{\operatorname{cov}(\mathbb{E}[X|Y])}{\mathbb{E}[\operatorname{cov}(X|Y)]} \geq 0$, larger means classes are more separated.

• For instance, if $\Pr(Y = 1) = \Pr(Y = -1) = 1/2$ and $X \mid Y = y \sim \mathcal{N}\left(y\mu, \sigma^2\right)$, for some $\mu \geq 0$ and $\sigma > 0$, we have

$$\mathbb{E}[\operatorname{cov}(X \mid Y)] = \sigma^2, \quad \operatorname{cov}(\mathbb{E}[X \mid Y]) = \mu^2$$

σ^2 therefore controls the within-class variance, and μ^2 the between-class variance.

Risk: $R(h) = \mathbb{E}[L(Y, h(X))] = R(h) = \begin{cases} \mathbb{E}\left[(Y - h(X))^2\right] & \text{under the squared loss} \\ \Pr(Y \neq h(X)) & \text{under the } 0-1 \text{ loss} \end{cases}$

Bayes Prediction Rule: $h^* = \arg\min_{h \in \mathcal{F}} R(h) = \arg\min_{h \in \mathcal{F}} R(h)$.

• For squared loss,

$$\begin{aligned} h^*(x) &= \mathbb{E}[Y \mid X = x] \\ R\left(h^*\right) &= \mathbb{E}_X[\operatorname{var}(Y \mid X)] \\ R(h) - R\left(h^*\right) &= \mathbb{E}_X\left[\left(h(X) - h^*(X)\right)^2\right] \end{aligned}$$

• For 0-1 loss,

$$\begin{aligned} h^*(x) = \arg\max_{k \in \mathcal{Y}} \Pr(Y = k \mid X = x) &= \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases} \quad \text{(binary)} \\ R\left(h^*\right) = 1 - \mathbb{E}_X\left[\max_{k \in \mathcal{Y}} \Pr(Y = k \mid X)\right] &= \frac{1}{2} - \mathbb{E}[|\eta(X) - 1/2|] \text{ (binary)} \\ R(h) - R\left(h^*\right) = \mathbb{E}_X\left[(2\eta(X) - 1)\left(1_{h^*(X)=1} - 1_{h(X)=1}\right)\right] & \end{aligned}$$

Empirical Risk Minimization: $\widehat{h}^{(d)} = \arg\min_{h \in \mathcal{H}} \widehat{R}_n(h) \implies \widehat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n L\left(y_i, h_\theta\left(x_i\right)\right)$

• The ultimate goal is to minimize the true loss rather than the empirical loss, but true loss is unknown as the distribution $P_{X,Y}$ is unknown.

Plug in Method: estimate the conditional distribution of Y given X, and set $\widehat{h}^{(d)}(x) = \arg\min_{y^* \in \mathcal{Y}} \int_{\mathcal{Y}} L\left(y, y^*\right) d\widehat{F}_x(y)$.

• Conditional plug-in directly estimates the conditional distribution:

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} \sum_{i=1}^n \log\left(f_\theta\left(y_i \mid x_i\right)\right)$$

• Generative plug-in first estimates the joint (X,Y), then estimate the conditional distribution by Bayes thm:

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} \sum_{i=1}^n \log\left(\pi_\theta\left(x_i, y_i\right)\right)$$

$$\implies \widehat{h}^{(d)}(x) = \begin{cases} \int_{\mathbb{R}} y f_{\widehat{\theta}}(y \mid x) dy \text{ for regression} \\ \underset{k=1,\ldots,K}{\arg\max} f_{\widehat{\theta}}(k \mid x) \text{ for classification} \end{cases}$$

2.2 Generative Classifiers

$$\begin{aligned} \widehat{h}^{(d)}(x) &= \underset{k\in\{1,\ldots,K\}}{\arg\max}\, \widehat{\pi}_k \widehat{g}_k(x), \text{ where } \pi_k = P(Y=k), g_k(x) = P(X|Y=k) \\ &= \underset{k\in\{1,\ldots,K\}}{\arg\max}\, \log\left(\widehat{\pi}_k\right) + \log\left(\widehat{g}_k(x)\right) \\ \implies \mathcal{L}\left(\pi_1,\ldots,\pi_K,\gamma\right) &= \sum_{k=1}^{K} m_k \log\left(\pi_k\right) - \gamma\left(\sum_{k=1}^{K}\pi_k - 1\right), \text{ where } m_k = \#\{j : y_j = k\} \\ \implies \hat{\pi}_k &= \frac{m_k}{n}, \text{ i.e. the MLE} \end{aligned}$$

To get $\hat{g_k}(x)$, seee the following 3 classifiers: **Linear Discriminant Analysis, Quadratic Discriminant Analysis, Naive Bayes**.

2.2.1 Linear Discriminant Analysis (LDA)

LDA assumes the class conditional densities are normal pdfs with a shared covariance matrix:

$$g_k(x) = P(X|Y=k) = \psi(x;\mu_k,\Sigma)$$

where $\mathcal{X} = \mathbb{R}^p$, μ_k $(k = 1, ..., p)$ is the centroid of the class (a vector of size p), and Σ is the within-class covariance matrix $(p \times p)$.
The loss function is:

$$\begin{aligned} &\ell\left(\pi_1,\ldots,\pi_K,\mu_1,\ldots,\mu_K,\Sigma\right) \\ &= \sum_{i=1}^{n}\left(\log\left(\pi_{y_i}\right) + \log\varphi\left(x_i;\mu_{y_i},\Sigma\right)\right) \\ &= \sum_{k=1}^{K}\left(m_k\log\left(\pi_k\right) + \sum_{i|y_i=k}\log\varphi\left(x_i;\mu_k,\Sigma\right)\right) \\ &= \left(\sum_{k=1}^{K} m_k\log\left(\pi_k\right)\right) - \frac{1}{2}\sum_{k=1}^{K}\sum_{i|y_i=k}\left(x_i-\mu_k\right)^\top\Sigma^{-1}\left(x_i-\mu_k\right) - \frac{n}{2}\log|\Sigma| + \text{constant} \end{aligned}$$

$$\implies \widehat{\pi}_k = \frac{m_k}{n}, \text{ under constraint } \sum_{k=1}^{K}\pi_k = 1$$

$$\implies \frac{\partial\ell}{\partial\mu_k} = \sum_{i|y_i=k}\Sigma^{-1}\left(x_i-\mu_k\right) = 0 \implies \widehat{\mu}_k = \frac{1}{m_k}\sum_{i:y_i=k} x_i$$

$$\implies \frac{\partial\ell}{\partial\Sigma} = \frac{1}{2}\sum_{k=1}^{K}\sum_{i|y_i=k}\Sigma^{-1}\left(x_i-\mu_k\right)\left(x_i-\mu_k\right)^\top\Sigma^{-1} - \frac{n}{2}\Sigma^{-1} = 0$$

$$\implies \widehat{\Sigma} = \frac{1}{n}\sum_{k=1}^{K}\sum_{i:y_i=k}\left(x_i-\widehat{\mu}_k\right)\left(x_i-\widehat{\mu}_k\right)^\top$$

Hence prediction rule is:

$$\begin{aligned}\widehat{h}^{(d)}(x) &= \underset{k=1,\ldots,K}{\arg\max} \log\left(\widehat{\pi}_k\right) - \frac{1}{2}\underbrace{\left(x-\widehat{\mu}_k\right)^\top \widehat{\Sigma}^{-1}\left(x-\widehat{\mu}_k\right)}_{\text{squared Mahalanobis distance}} \\ &= \underset{k=1,\ldots,K}{\arg\max} \underbrace{\log\left(\widehat{\pi}_k\right) - \frac{1}{2}\widehat{\mu}_k^\top \widehat{\Sigma}^{-1}\widehat{\mu}_k}_{\widehat{a}_k} + \underbrace{\widehat{\mu}_k^\top \widehat{\Sigma}^{-1}}_{\widehat{b}_k^\top} x, \text{ the linear discriminant function}\end{aligned}$$

Assuming $\hat{\Sigma}$ has full rank, then the eigen decomposition gives $\hat{\Sigma} = V\Lambda V^T$, where V is $p \times p$ orthogonal matrix and Λ is $p \times p$ diagonal matrix. Note $\hat{\Sigma}^{-1} = V\Lambda^{-1}V^T$ and $\hat{\Sigma}^{-1/2} = \Lambda^{-1/2}V^T$. Then,

$$\begin{aligned}\left(x-\widehat{\mu}_k\right)^\top \widehat{\Sigma}^{-1}\left(x-\widehat{\mu}_k\right) &= \left(\Lambda^{-1/2}V^\top x - \Lambda^{-1/2}V^\top \widehat{\mu}_k\right)^\top \left(\Lambda^{-1/2}V^\top x - \Lambda^{-1/2}V^\top \widehat{\mu}_k\right) \\ &= \left\|x^\bullet - \widehat{\mu}_k^\bullet\right\|^2 \text{ , Euclidean distance between the transformed vectors}\end{aligned}$$

where $x^\bullet = \Lambda^{-1/2}V^T x, \hat{\mu}_k^\bullet = \Lambda^{-1/2}V^T\hat{\mu}_k$.

Dimentionality Reduction:
Estimated between-class covariance : $\widehat{B} = \widehat{Cov(E(X|Y))} = \sum_{k=1}^K \widehat{\pi}_k\left(\widehat{\mu}_k - \widehat{\mu}\right)\left(\widehat{\mu}_k - \widehat{\mu}\right)^\top$
Transformed between-class covariance: $\widehat{B}^\bullet = \widehat{Cov}(E(\Sigma^{-1/2}X|Y)) = \sum_{k=1}^K \widehat{\pi}_k\left(\widehat{\mu}_k^\bullet - \widehat{\mu}^\bullet\right)\left(\widehat{\mu}_k^\bullet - \widehat{\mu}^\bullet\right)^\top$, where $\hat{\mu}^\bullet = \sum_{k=1}^K \pi_k\hat{\mu}_k^\bullet$. Denote $\hat{B}^\bullet = U^\bullet D^\bullet U^{\bullet T}$, with u_l being the l'th column of $U^\bullet$.
The l'th discriminant coordinate of x is:

$$z_\ell = \left(u_\ell^\bullet\right)^\top x^\bullet = \left(u_\ell^\bullet\right)^\top \Lambda^{-1/2}V^\top x$$

Note that $a_1 = V\Lambda^{-1/2}u_1^\bullet$ maximizes the estimated separability: $\frac{a_1 B a_1^T}{a_1 \Sigma a_1^T} = \frac{u_1 B u_1^T}{u_1 u_1^T}$, hence $a_1^T x$ is the one-dimensional projection that maximizes the separation between classes.

2.2.2 Quadratic Discriminant Analysis (QDA)

Assuming the class conditional densities $g_k(x)$ are normal pdfs with unconstrained covariances:

$$g_k(x) = \varphi\left(x; \mu_k, \Sigma_k\right)$$

The loss function is similiar to the LDA's:

$$\begin{aligned}&\ell\left(\pi_1,\ldots,\pi_K,\mu_1,\ldots,\mu_K,\Sigma_1,\ldots\Sigma_K\right) \\ &= \left(\sum_{k=1}^K m_k \log\left(\pi_k\right)\right) - \frac{1}{2}\sum_{k=1}^K \sum_{i|y_i=k}\left(x_i-\mu_k\right)^\top \Sigma^{-1}\left(x_i-\mu_k\right) - \sum_{k=1}^K \frac{m_k}{2}\log|\Sigma| + \text{constant} \\ &\implies \widehat{\Sigma}_k = \frac{1}{m_k}\sum_{i:y_i=k}\left(x_i-\widehat{\mu}_k\right)\left(x_i-\widehat{\mu}_k\right)^\top\end{aligned}$$

The quadratic decision boundary is:

$$\begin{aligned}\widehat{h}^{(d)}(x) &= \underset{k=1,\ldots,K}{\arg\max} \log\left(\widehat{\pi}_k\right) - \frac{1}{2}\log\left|\widehat{\Sigma}_k\right| - \frac{1}{2}\underbrace{\left(x-\widehat{\mu}_k\right)^\top \widehat{\Sigma}_k^{-1}\left(x-\widehat{\mu}_k\right)}_{\text{squared Mahalanobis distance}} \\ &= \underset{k=1,\ldots,K}{\arg\max} \log\left(\widehat{\pi}_k\right) - \frac{1}{2}\log\left|\widehat{\Sigma}_k\right| - \frac{1}{2}\widehat{\mu}_k^\top \widehat{\Sigma}_k^{-1}\widehat{\mu}_k + \widehat{\mu}_k^\top \widehat{\Sigma}_k^{-1}x - \frac{1}{2}x^\top \widehat{\Sigma}_k^{-1}x.\end{aligned}$$

Algorithm 4: LDA & QDA

Training:

• Compute the MLE: $\widehat{\pi}_k = \frac{m_k}{n}, \widehat{\mu}_k = \frac{1}{m_k}\sum_{i:y_i=k} x_i$
and $\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{K}\sum_{i:y_i=k}(x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^\top$ (LDA) or
$\widehat{\Sigma}_k = \frac{1}{m_k}\sum_{i:y_i=k}(x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^\top$ (QDA);

• Compute the eigen decomposition $\hat{\Sigma} = V\Lambda V^T$;

• For $k = 1, \ldots, K,$ set the transformed centroid$\widehat{\mu}_k^\bullet = \Lambda^{-1/2}V^\top\widehat{\mu}_k$

Prediction:

for *x in input X* **do**

- Transform $x^\bullet = \Lambda^{-1/2}V^T x$
- Classify to the closest class mean $\hat{\mu}_k$, while trying to maximize the class proportion $\hat{\pi}_k$:

$$\widehat{h}^{(d)}(x) = \underset{k=1,\ldots,K}{\arg\min} \|x^\bullet - \widehat{\mu}_k^\bullet\|^2 - 2\log(\widehat{\pi}_k) \text{ (LDA)}$$

end

Shrinkage: (to prevent overfitting of LDA)

Replace the covariance matrix with $\hat{\Sigma}(\delta) = \delta I_p + (1-\delta)\hat{\Sigma}$, for $\delta \in (0,1)$

Regularized Discriminant Analysis: (LDA + QDA)

$\widehat{\Sigma}_k(\alpha) = \alpha\widehat{\Sigma}_k + (1-\alpha)\widehat{\Sigma}$

2.2.3 Naive Bayes

The "Naive" assumption: all features are independent conditioned on the class labels, i.e. $g_k(x) = \prod_{j=1}^{p} g_{kj}(x_j;\theta_{kj}), \forall j = 1, ..., p.$

3 Standard Choice of parametric models:

- $g_{kj}(x_j;\theta_{kj}) = \varphi\left(x_j;\mu_{kj},\sigma_{kj}^2\right)$, Gaussian model for real-valued features;
- $g_{kj}(1;\theta_{kj}) = 1 - g_{kj}(0;\theta_{kj})$, Bernoulli model for binary features;
- $g_{kj}(c;\theta_{kj}) = \theta_{kj,c}(x_j \in \{1, ..., C\})$, Multinomial model for categorical features

Parameter Estimation:

The joint log-likelihood is a summation, given the independence assumption:

$$\begin{aligned}
\ell\left((\pi_k)_{k=1\ldots,K}, (\theta_{kj})_{k=1,\ldots,K;j=1,\ldots,p}\right) &= \sum_{i=1}^{n}\left(\log(\pi_{y_i}) + \log g_{y_i}(x_i)\right) \\
&= \sum_{k=1}^{K} m_k \log(\pi_k) + \sum_{k=1}^{K}\sum_{i|y_i=k}\log g_k(x_i) \\
&= \sum_{k=1}^{K} m_k \log(\pi_k) + \sum_{k=1}^{K}\sum_{i|y_i=k}\sum_{j=1}^{p}\log g_{kj}(x_{ij};\theta_{kj}) \\
&= \underbrace{\sum_{k=1}^{K} m_k \log(\pi_k)}_{\ell_1((\pi_k))} + \sum_{j=1}^{p}\sum_{k=1}^{K}\underbrace{\sum_{i|y_i=k}\log g_{kj}(x_{ij};\theta_{kj}))}_{\ell_{kj}(\theta_{kj})}
\end{aligned}$$

As before, $\hat{\pi}_k = \frac{m_k}{n}$ under the regularization that π_k sums to 1.
For the θ's: $\widehat{\theta}_{kj} = \arg\max_{\theta_{kj}} \sum_{i|y_i=k} \log g_{kj}\left(x_{ij};\theta_{kj}\right)$

$$\text{Gaussian}: \widehat{\mu}_{kj} = \frac{1}{m_k}\sum_{i|y_i=k} x_{ij}, \quad \widehat{\sigma}^2_{kj} = \frac{1}{m_k}\sum_{i|y_i=k}\left(x_{ij}-\widehat{\mu}_{kj}\right)^2$$

$$\text{Bernoulli}: \widehat{\theta}_{kj} = \frac{\sum_{i|y_i=k} x_{ij}}{m_k}$$

$$\text{Multinomial}: \widehat{\theta}_{kj,c} = \frac{\sum_{i|y_i=k} \mathbb{1}_{x_{ij}=c}}{m_k}$$

The conditional distribution: $P(Y=k|X=x) = \frac{\widehat{\pi}_k \prod_{j=1}^p g_{kj}\left(x;\widehat{\theta}_{kj}\right)}{\sum_{k'=1}^K \widehat{\pi}_{k'} \prod_{j=1}^p g_{k'j}\left(x;\hat{\theta}_{k'j}\right)}$.
The naive Bayes classifier: $\widehat{h}^{(d)}(x) = \arg\max_{k\in\{1,\ldots,K\}} \log\left(\widehat{\pi}_k\right) + \sum_{j=1}^p \log\left(g_{kj}\left(x;\widehat{\theta}_{kj}\right)\right)$.

Algorithm 5: Naive Bayes

Training:
Determine the corresponding model assumptions matching the features:

- $g_{kj}\left(x_j;\theta_{kj}\right) = \varphi\left(x_j;\mu_{kj},\sigma^2_{kj}\right)$, Gaussian model for real-valued features;
- $g_{kj}\left(1;\theta_{kj}\right) = 1 - g_{kj}\left(0;\theta_{kj}\right)$, Bernoulli model for binary features;
- $g_{kj}\left(c;\theta_{kj}\right) = \theta_{kj,c}(x_j \in \{1,\ldots,C\})$, Multinomial model for categorical features

Compute the MLE: $\widehat{\mu}_{kj} = \frac{1}{m_k}\sum_{i|y_i=k} x_{ij}, \quad \widehat{\sigma}^2_{kj} = \frac{1}{m_k}\sum_{i|y_i=k}\left(x_{ij}-\widehat{\mu}_{kj}\right)^2, \widehat{\theta}_{kj} = \frac{\sum_{i|y_i=k} x_{ij}}{m_k}, \widehat{\theta}_{kj,c} = \frac{\sum_{i|y_i=k} \mathbb{1}_{x_{ij}=c}}{m_k}$
Prediction: Classify the test input x by the NBClassifier:
$\widehat{h}^{(d)}(x) = \arg\max_{k\in\{1,\ldots,K\}} \log\left(\widehat{\pi}_k\right) + \sum_{j=1}^p \log\left(g_{kj}\left(x;\widehat{\theta}_{kj}\right)\right)$
Missing Data: automatically marginalize out the missing data
$\widehat{h}^{(d)}(x) = \arg\max_{k\in\{1,\ldots,K\}} \log\left(\widehat{\pi}_k\right) + \sum_{j \text{ not missing}} \log\left(g_{kj}\left(x;\widehat{\theta}_{kj}\right)\right)$

2.2.4 Summary of Generative Classifiers

Pros:
The assumptions made the data generating process can be statistically tested, e.g. Gaussality, independence;
The generative classifiers make good prediction in practice even assumptions do not hold;
Generative classifiers can easily handle missing data in the input;
Interpretable predictions;
Cons:
model assumptions are rarely met in practice.

2.3 Key Concepts in SML

2.3.1 Nonlinear Input Transformation/Expansion

Multivariate Linear Regression: input/target data is $\{(x_i,y_i)\}_{i=1}^n, x_i \in \mathbb{R}^p$. The class of linear

prediction rule is: $\mathcal{H} = \left\{h(x) = \widetilde{x}^\top \beta = \begin{pmatrix} 1 & x^\top \end{pmatrix} \beta \mid \beta \in \mathbb{R}^{p+1}\right\}$.
• The ERM under the squared loss is: $\hat{\beta} = \underset{\beta \in \mathbb{R}^{p+1}}{\arg\min} \|\mathbf{y} - \tilde{\mathbf{X}}\beta\|^2 = (\tilde{\mathbf{X}}^\mathbf{T}\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\mathbf{T}\mathbf{y}$, where:

$$\widetilde{\mathbf{X}} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix}$$

Non-Linear Input Expansion: $\mathcal{H} = \left\{h(x) = \phi(\widetilde{x})^\top \beta \mid \beta \in \mathbb{R}^{p'}\right\}$, where p' can be either $< p$ (dimension reduction) or $> p$ (input expansion).

2.3.2 Overfitting and Bias Variance Trade-off

Estimation-Approximation error:

$$\underbrace{R\left(\widehat{h}^{(d)}\right) - R\left(h^*\right)}_{\text{excess risk}} = \underbrace{R\left(\widehat{h}^{(d)}\right) - R\left(h^*_{\mathcal{H}}\right)}_{\text{estimation error}} + \underbrace{R\left(h^*_{\mathcal{H}}\right) - R\left(h^*\right)}_{\text{approximation error}}$$

• *Approximation error* is the difference between the risk of the best prediction rule within the class and the Bayes risk. Large hypothesis class $\implies$ complex prediction rule $\implies$ smaller approx error.
• *Estimation error* is the difference between the risk of the learned prediction rule and of the best prediction rule in the class. Large hypothesis class $\implies$ more parameters to estimate $\implies$ larger est error.
• $n_{train} \to \infty \implies$ est error $\to 0 \implies$ excess risk $\to$ approx error, i.e. $R\left(\widehat{h}^{(d)}\right) \to R\left(h^*_{\mathcal{H}}\right)$
• $\mathbb{E}\left[\widehat{R}_n^{(D)}\left(\widehat{h}^{(D)}\right)\right] \leq R\left(h^*_{\mathcal{H}}\right)$ for random sample D proof?, i.e. empirical risk of the learned rule $\hat{R}_n(\hat{h})$ is lower than the true risk $R(h^*_{\mathcal{H}})$ of that prediction rule. The generalization gap $R(\hat{h}) - \hat{R}_n(\hat{h}) \to 0$, as they both converges to the true risk.

Bias-variance decomposition: under the squared loss,

$$\begin{aligned}
&\mathbb{E}_D\left[R\left(\widehat{h}^{(D)}\right) - R\left(h^*\right)\right] \\
&= \mathbb{E}_D\mathbb{E}_X\left[\left(\widehat{h}^{(D)}(X) - h^*(X)\right)^2\right] \\
&= \mathbb{E}_X\mathbb{E}_D\left[\left(\widehat{h}^{(D)}(X) - h^*(X)\right)^2\right], \text{let } \bar{h}_{\mathcal{H}} = \mathbb{E}_D[\hat{h}^{(D)}(x)] \\
&= \mathbb{E}_X\mathbb{E}_D\left[\left(\widehat{h}^{(D)}(x) - \bar{h}_{\mathcal{H},n}(x)\right)^2 + \left(\bar{h}_{\mathcal{H},n}(x) - h^*(x)\right)^2 + 2\left(\widehat{h}^{(D)}(x) - \bar{h}_{\mathcal{H},n}(x)\right)\left(\bar{h}_{\mathcal{H},n}(x) - h^*(x)\right)\right] \\
&= \underbrace{\mathbb{E}_X\left[\mathbb{E}_D\left(\widehat{h}^{(D)}(x) - \bar{h}_{\mathcal{H},n}(x)\right)^2\right]}_{\mathbb{E}_X[v_{\mathcal{H},n}(X)]} + \underbrace{\mathbb{E}_X\left[\mathbb{E}_D\left(\bar{h}_{\mathcal{H},n}(x) - h^*(x)\right)^2\right]}_{\mathbb{E}_X[b_{\mathcal{H},n}(X)^2]}
\end{aligned}$$

• Variance term $\propto$ estimation error, large for large hypothesis space $\mathcal{H}$ and vanishes to zero for large dataset.

- The variance of an algorithm refers to the amount of variation that is observed between models that are obtained by training an algorithm (using a given set of hyperparameter) on different data sets (of a given size) sampled from an underlying data generating distribution.

• Bias term $\propto$ approximation error, small for large hypothesis space $\mathcal{H}$ and does not vanish to zero for large dataset.
- The bias of an algorithm refers to the difference between the true data model and the average model obtained by training an algorithm (using a given set of hyperparameter) on different data sets (of a given size) sampled from an underlying data generating distribution.

2.3.3 Regularized ERM

$$\widehat{h}^{(d)} = \underset{h\in\mathcal{H}}{\arg\min} \left\{ \widehat{R}_n(h) + \frac{\lambda}{n} \operatorname{pen}(h) \right\}$$

where $\lambda > 0$ is the regularization parameter.

If the prediction rule $h \in \mathcal{H}$ is parametrized by $\beta = [\beta_0, \cdots, \beta_M] \in \mathbb{R}^{M+1}$, then:
(i) **Tikhonov regularisation (L_2 penalty)**: $pen(h) = \|\beta\|^2 = \sum_{j=0}^{M} \beta_j^2$.
Ridge regression estimator: $\widehat{\beta} = \underset{\beta\in\mathbb{R}^{M+1}}{\arg\min} \sum_{i=1}^{n} \left(y_i - \phi\left(x_i\right)^\top \beta \right)^2 + \lambda\|\beta\|^2 = \left(\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda I\right)^{-1}\mathbf{\Phi}^\top\mathbf{y}$
•$n \to \infty \implies \frac{\lambda}{n}\operatorname{pen}(h) \to 0 \implies \hat{R}_n(h) \overset{a.s}{\to} R(h)$, effect of regularization vanishes as we have more data.
(ii) L_1 **LASSO penalty**: $pen(h) = \|\beta\|_1 = \sum_{j=0}^{M} |\beta_j|$.
(iii) **Elastic Net ($L_1 + L_2$)**: $pen(h) = \delta\|\beta\|_1 + (1-\delta)\|\beta\|_2^2$ (iv) **Sobolev norm**: $\|f\|_{W^1}^2 = \int_{-\infty}^{+\infty} f(x)^2 dx + \int_{-\infty}^{+\infty} f'(x)^2 dx$
• Penalize some notion of smoothness of the function (e.g. large derivatives).

2.3.4 Cross-Validation

Algorithm 6: Training-Validation-Test

Training:

for $j = 1, ..., M_{\max}$ **do**

Estimate the learned prediction rule using the training set d_{train}:

$$\widehat{h}^{(d_{\text{train}})} = \arg\min_{h \in \mathcal{H}_j} \hat{R}^{(d_{\text{train}})}(h)$$

end

Validation:

for $j = 1, ..., M_{\max}$ **do**

Compute $\hat{R}^{(d_{\text{val}})}(\hat{h}_j^{(d_{\text{train}})})$

end

Report the hypothesis class of the best prediction rule $\hat{M} = \arg\min_j \hat{R}^{(d_{\text{val}})}(\hat{h}_j^{(d_{\text{train}})})$

Re-train the prediction rule on both the training and validation set for the best hypothesis class $\hat{M}$:

$$\widehat{h}^{(d_{\text{train}}, d_{\text{val}})} = \arg\min_{h \in \mathcal{H}_{\widehat{M}}} R^{(d_{\text{train}}, d_{\text{val}})}(h)$$

Testing:

Approximate the generalisation error: $\hat{R}^{(d_{\text{test}})}(\hat{h}_j^{(d_{\text{train, val}})})$

Remarks:

• Cannot use the empirical risk from the training set $\hat{R}^{d_{\text{train}}}(\hat{h}_j^{d_{\text{train}}})$ to estimate the true risk $R(\hat{h}_j^{d_{\text{train}}})$, should instead use the empirical risk estimated from the validation set:

$$\mathbb{E}_{D_{\text{val}}}\left[\widehat{R}^{(D_{\text{val}})}\left(\widehat{h}_j^{(D_{\text{train}})}\right) \mid D_{\text{train}} = d_{\text{train}}\right] = R\left(\widehat{h}_j^{(d_{\text{train}})}\right)$$

(This is an unbiased estimator of the jth learned prediction rule.)

• Unbiased estimate of the true risk of the learned prediction rule, where $\widehat{h}^{(d_{\text{train}}, d_{\text{val}})}$ is the best class prediction rule $\hat{h}_{\hat{M}}^{(d_{\text{train}})}$ retrained over the train and validation set:

$$\mathbb{E}\left[\widehat{R}^{(D_{\text{test}})}\left(\widehat{h}^{(D_{\text{train}}, D_{\text{val}})}\right) \mid D_{\text{train}} = d_{\text{train}}, D_{\text{val}} = d_{\text{val}}\right] = R\left(\widehat{h}^{(d_{\text{train}}, d_{\text{val}})}\right)$$

Algorithm 7: K-Fold Cross Validation

Initialize the number of folds T and split the train/validation set into T folds.

Training:

for $t = 1, ..., T$ **do**

  Use fold t as the validation set and the rest as the training set.

  for $j = 1, ..., M_{\max}$ **do**

    Train the prediction rule $\hat{h}_j^{(d_{\text{train, t}})}$ and compute the estimated risk $\hat{R}_j^{(d_{\text{train, t}})}$

  end

end

Validation:

• Choose the learner that minimises the estimated risk average over the folds:

$$\widehat{M} = \arg\min_j \frac{1}{T} \sum_{t=1}^{T} \widehat{R}^{(d_{\text{val},t})} \left(\widehat{h}_j^{(d_{\text{train},t})} \right)$$

• Retrain the prediction rule on the whole train/validation set.

Testing:

Approximate the generalisation error with $\hat{R}^{(d_{\text{test}})}(\hat{h}^{(d_{\text{train/val}})})$

2.3.5 Evaluations of Binary Classification

Confusion Matrix:

	h(x) = -1	h(x) = 1
y = -1	true negative	false positive (Type I error)
y = 1	false negative (Type II error)	true positive

Empirical Misclassification Error: $\widehat{R}_n^{(d)}(h) = \frac{FN+FP}{FN+FP+TP+TN}$

Performance Assessments:

Name	Population	Empirical
Misclassification error (Error rate)	$Pr(Y \neq h(X))$	(FP + FN)/(TP + TN + FP + FN)
Accuracy (1-misclassification)	$Pr(Y = h(X))$	(TP + TN)/(TP + TN + FP + FN)
Sensitivity/Recall/True positive rate	$Pr(h(X) = 1 \mid Y = 1)$	TP/(TP + FN)
Specificity/True negative rate	$Pr(h(X) = -1 \mid Y = -1)$	TN/(TN + FP)
False positive rate (1-specificity)	$Pr(h(X) = 1 \mid Y = -1)$	FP/(TN + FP)
Precision	$Pr(Y = 1 \mid h(X) = 1)$	TP/(TP + FP)

Weighted Loss: assign different cost to different types of errors $L(y, h(x)) = a1_{y=-1,h(x)=1} + b1_{y=1,h(x)=-1}$

$\implies$ Bayes Classifier $h_t^*(x) = \begin{cases} 1, \text{, if } \eta(x) \geq t := \frac{a}{a+b} \\ -1, o/w \end{cases}$, where $\eta(x) = Pr(Y = 1 \mid X = x)$.

($t = 1/2$ gives the Bayes classifier under the 0-1 loss, large t $\implies$ more false negative and less false positive, vice versa.)

ROC and AUC: Denote $\alpha(t) = \Pr\left(\widehat{h}_t(X) = 1 \mid Y = -1\right)$ as FPR and $\beta(t) = \Pr\left(\widehat{h}_t(X) = 1 \mid Y = 1\right)$ as TPR, then $\alpha(1) = \beta(1) = 0$ and $\alpha(0) = \beta(0) = 1$, and:

$$AUC = \int_1^0 \beta(t) d\alpha(t)$$

• AUC is a measure of the performance of the family of plug-in classifiers over the whole range of loss functions.
• **Probabilistic Interpretation of AUC**: $AUC = \Pr\left(\widehat{\eta}\left(\tilde{X}_1\right) \geq \widehat{\eta}\left(\tilde{X}_0\right) \mid \tilde{Y}_1 = 1, \tilde{Y}_0 = -1, D = d\right)$, where $\widehat{\eta}(x)$ is the estimated conditional probability ($P(Y = 1|X = x)$) of a plug-in classifier, and $(\tilde{X}_0, \tilde{Y}_0), (\tilde{X}_1, \tilde{Y}_1)$ are mutually independent RVs independent of the training set D.
proof: For a random realization of the training set D = d, denote the cdfs:

$$\begin{aligned}
F_1(s) &= \Pr(\widehat{\eta}(X) \leq s \mid Y = 1) \implies \beta(t) = 1 - F_1(t) \\
F_{-1}(s) &= \Pr(\widehat{\eta}(X) \leq s \mid Y = -1) \implies \alpha(t) = 1 - F_{-1}(t) \\
\implies \int_1^0 \beta(t) d\alpha(t) &= \int_0^1 (1 - F_1(t)) f_{-1}(t) dt = \mathbb{E}\left[\left(1 - F_1\left(\widehat{\eta}\left(\tilde{X}_0\right)\right)\right) \mid \tilde{Y}_0 = -1\right] \\
&= \mathbb{E}\left[\Pr\left(\widehat{\eta}\left(\widetilde{X}_1\right) \geq \widehat{\eta}\left(\widetilde{X}_0\right) \mid \widetilde{Y}_1 = 1, \widetilde{X}_0\right) \mid \widetilde{Y}_0 = -1\right] \\
&= \Pr\left(\widehat{\eta}\left(\widetilde{X}_1\right) \geq \widehat{\eta}\left(\widetilde{X}_0\right) \mid \widetilde{Y}_1 = 1, \widetilde{Y}_0 = -1\right) \\
\overset{\text{Or}}{\implies} \int_1^0 \beta(t) d\alpha(t) &= \int_0^1 (1 - F_1(t)) f_{-1}(t) dt = \int_0^1 \int_0^1 1_{s \geq t} f_1(s) ds f_{-1}(t) dt \\
&= \mathbb{E}\left[1_{\widehat{\eta}(\tilde{X}_1) \geq \widehat{\eta}(\tilde{X}_0)} \mid \widetilde{Y}_0 = -1, \widetilde{Y}_1 = 1\right] \\
&\quad \because f_1(s) f_{-1}(t) \text{ is the joint cond. pdf of } (\hat{\eta}(\tilde{X}_1), \hat{\eta}(\tilde{X}_0)) \text{ given } (\tilde{Y}_1, \tilde{Y}_0) \\
&= \Pr\left(\widehat{\eta}\left(\tilde{X}_1\right) \geq \widehat{\eta}\left(\tilde{X}_0\right) \mid \tilde{Y}_0 = -1, \tilde{Y}_1 = 1\right)
\end{aligned}$$

2.3.6 Optimization

The object function we are aimed to minimizing:

$$\begin{aligned}
J(\theta) &= \sum_{i=1}^n J_i(\theta) + \underset{\text{penalty}}{J_0(\theta)} \\
&= \begin{cases} \frac{1}{n} \sum_{i=1}^n L(y_i, h_\theta(x_i)), \text{ for ERM} \\ \sum_{i=1}^n \log f_\theta(y_i|x_i), \text{ for conditional plug-in} \\ \sum_{i=1}^n \log \pi_\theta(x_i, y_i), \text{ for generative plug-in} \end{cases} + J_0(\theta)
\end{aligned}$$

The gradient and Hessian of J_θ: $\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{d\theta_1} \\ \cdots \\ \frac{\partial J(\theta)}{d\theta_p} \end{pmatrix}$, $\nabla^2_\theta J(\theta) = \begin{pmatrix} \frac{\partial^2 J(\theta)}{d\theta_1 d\theta_1} & \cdots & \frac{\partial^2 J(\theta)}{d\theta_1 d\theta_p} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 J(\theta)}{d\theta_p d\theta_1} & \cdots & \frac{\partial^2 J(\theta)}{d\theta_p d\theta_p} \end{pmatrix}$

positive semi-definite: symmetric $p \times p$ real-valued matrix $H \succeq 0$ iff $z^T H z \geq 0, \forall z \in \mathbb{R}^p$.
Convex function: $J(\alpha u + (1 - \alpha)v) \leq \alpha J(u) + (1 - \alpha) J(v)$, e.g.
- Univariate: $\theta^2, \exp(-\theta), \log(1 + \exp(-\theta)), \max(0, 1 - \theta)$
- Affine functions: $A\theta + b$
- Quadratic functions: $\theta^\top H \theta$ where $H \succeq 0$

Properties of Convex functions:
(i) If the function J is convex, all local minima are also global minima;
(ii) A differentiable function J is convex iff $J(u) \geq J(v) + \nabla J(v)^T(u-v), \forall u, v$;
(iii) $\nabla_u J(u) = 0 \implies$ gloabal minimum at u;
(iv) $\nabla_u^2 J(u) \succeq 0 \iff J$ is convex, for twice differentiable function;
(v) Non-negative linear combination ($J = \alpha_1 J_1 + \alpha_2 J_2$) and affine composition ($J(\theta) = g(A\theta + b)$) are convex.

Gradient-Based Optimization Methods: GD/SGD/N-R

Algorithm 8: Gradient Optimization Methods

Initialize θ_0 and set t = 0.
Initialize learning rate $\eta > 0$, tolerance $\epsilon > 0$, mini-batch size n_b
while *not converge* **do**
  Compute the gradient: $\nabla_\theta J(\theta) = \frac{1}{n}\left(\sum_{i=1}^n \nabla_\theta L\left(y_i, h_\theta\left(x_i\right)\right) + \lambda \nabla_\theta \operatorname{pen}\left(h_\theta\right)\right)$
  Compute the Hessian: $\nabla_\theta^2 J(\theta)$
  Randomly sample n_b observation $(\tilde{x_i}^{(t)}, \tilde{y_i}^{(t)})$ from the dataset, and compute the gradient estimate: $\nabla_\theta \tilde{J}^{(t)}(\theta) = \frac{1}{n_b}\left(\sum_{i=1}^{n_b} \nabla_\theta L\left(\tilde{y_i}^{(t)}, h_\theta\left(\tilde{x_i}^{(t)}\right)\right) + \lambda \nabla_\theta \operatorname{pen}\left(h_\theta\right)\right)$
  Update Step:
  $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta J\left(\theta^{(t)}\right)$, Gradient Descent
  $\theta^{(t+1)} = \theta^{(t)} - (\nabla_\theta^2 J(\theta^{(t)}))^{-1} \nabla_\theta J\left(\theta^{(t)}\right)$, Newton Raphson
  $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \tilde{J}^{(t)}\left(\theta^{(t)}\right)$, Stochastic Gradient Descent
end
Early Stopping (optional):
for $t = 0, 1, ..., t_{\max}$ **do**
  Approximate validation risk $R(\theta^{(t)})$ and choose $\theta^{(t)} = \underset{t=0,1,...,t_{\max}}{\arg\min}\ \hat{R}(\theta^{(t)})$
end

Scheduler: Slowly decrease the learning rate over the training loop to ensure convergence, by ensuring that the **Robbins–Monro** conditions are satisfied: $\sum_{t=1}^\infty \eta_t = \infty$ and $\sum_{t=1}^\infty \eta_t^2 < \infty$.
• A good rule of thumb: find the smallest step size such that SGD becomes unstable, then reduce it by a factor of 2-10. Alternatively, find the largest stable step size.

Momentum: Average out the stochastic gradients over previous minibatches, so that larger learning rates can be used (with effectively the same variance, the convergence can be faster).

$$\Delta_t = \beta \Delta_{t-1} + (1-\beta)\widehat{\nabla_\theta \hat{R}}\left(\theta_t\right), \quad 0 < \beta < 1$$
$$\theta_t = \theta_{t-1} - \eta_t \Delta_t$$

ADAM: Update different parameters (may of different scales) with different learning rates, by keeping weighted moving average estimates for both the gradient Δ_t and the squared gradient V_t,

scaling the step size using the latter (which is related to the curvature of the loss landscape):

$$\Delta_t = \beta_1 \Delta_{t-1} + (1-\beta_1)\,\widehat{\nabla_\theta \hat{R}}\,(\theta_{t-1}) \qquad \tilde{\Delta}_t = \frac{\Delta_t}{1-(\beta_1)^t}$$

$$V_t = \beta_2 V_{t-1} + (1-\beta_2)\left(\widehat{\nabla_\theta \hat{R}}\,(\theta_{t-1})\right)^2 \quad \tilde{V}_t = \frac{V_t}{1-(\beta_2)^t}$$

$$\theta_t = \theta_{t-1} - \frac{\alpha_t}{\sqrt{\tilde{V}_t}+\epsilon}\tilde{\Delta}_t$$

where $0 < \beta_1, \beta_2 < 1$, $\Delta_0 = V_0 = 0$.
• Dividing $1-\beta^t$ ensures that the exponentially decaying weights for averaging sum to 1 (i.e. it accounts for the fact that the earlier steps are averaging over fewer gradients because of the zero initialization).
• Often choose large β_1, β_2 (0.99, 0.999 respectively), such that large amounts of historical gradient information is accumulated and the gradients change quite slowly.

Duality in Convex Optimization: want to solve the **primal problem**, i.e. minimize $f_0(x)$ subject to $f_i(x) \le 0, i = 1, \cdots, m$ and $h_j(x) = 0, j = 1, \cdots, r$.
• **Lagrangian**: $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}$

$$L(x, \lambda, \nu) := f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{r} \nu_j h_j(x)$$

• **Dual function**: $g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu)$, a pointwise infimum of affine functions of the **dual feasible pairs** (λ, ν) $(\lambda \ge 0, (\lambda, \nu) \in domain(g))$, hence it is <u>concave</u> in (λ, ν).
- When $\lambda \ge 0$, then

$$\begin{aligned} g(\lambda, \nu) &:= \inf_{x \in \mathcal{D}} \left(f_0(x) + \underbrace{\sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{r} \nu_j h_j(x)}_{\le 0} \right) \\ &\le f_0(\tilde{x}) + \sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{j=1}^{r} \nu_j h_j(\tilde{x}) \\ &\le f_0(\tilde{x}) = p^* \end{aligned}$$

• **Dual problem**: maximize $g(\lambda, \nu)$ subject to $\lambda \ge 0$, as $g(\lambda, \nu)$ is a lower bound of the Lagrangian.
- **weak duality**: $d^* \le p^*$, where $d^* = \max g(\lambda, \nu)$; **strong duality**: $d^* = p^*$; **optimal duality gap** $= p^* - d^*$
- **constrain qualifications** are the conditions under which strong duality holds:
(i) if the primal problem is convex, of the form: minimize $f_0(x)$ subject to $f_i(x) \le 0, i = 1, ..., n$ and $Ax = b$; and
(ii) if **Slater's condition** holds: $\exists$ strictly feasible point $\tilde{x} \in D$ s.t. $f_i(\tilde{x}) < 0, i = 1, ..., m, A\tilde{x} = b$. (The weaker version of Slater's condition is sufficient: $f_i(\tilde{x}) \le 0 (i = 1 : k), f_j(\tilde{x}) < 0 (j = k+1, ..., n), A\tilde{x} = b$).
- **max-min inequality**: $d^* = \sup_{\lambda \succeq 0} \inf_x L(x, \lambda) \le \inf_x \sup_{\lambda \succeq 0} L(x, \lambda) = p^*$

KKT Conditions: If a convex optimization problem with differentiable objective and constraint functions satisfies Slater's conditions, then the KKT conditions are necessary and sufficient for global optimality.

$$
\begin{aligned}
&f_i(x) \leq 0, i = 1, \ldots, m \\
&h_i(x) = 0, i = 1, \ldots, r \\
&\lambda_i \geq 0, i = 1, \ldots, m \\
&\lambda_i f_i(x) = 0, i = 1, \ldots, m, \textbf{ complementary slackness} \\
&\nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{r} \nu_i \nabla h_i(x) = 0.
\end{aligned}
$$

2.4 Linear Classifiers

Consider binary classification. The empirical risk minimizer under 0-1 loss is:

$$
\begin{aligned}
\widehat{\beta} &= \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} 1_{y_i \neq h(x_i)} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} 1_{y_i \neq \phi(x_i)^\top \beta} \\
&= \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} 1_{sign(yf(x_i))=-1} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \psi_{0-1}(yf(x))
\end{aligned}
$$

2.4.1 Surrogate loss function

continuous function s.t. convex and $\psi(x) \geq \psi_{0-1}(x) \forall x$.

$\implies$ ERM prediction rule: $\hat{h}(x) = \begin{cases} 1, \text{ if } \hat{f}(x) = \arg\min_{f \in \mathcal{H}_f} \frac{1}{n} \sum_{i=1}^{n} \psi\left(y_i f\left(x_i\right)\right) \geq 0 \\ -1, o/w \end{cases}$

$\implies$ estimate: $\widehat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \psi\left(y_i \phi\left(x_i\right)^\top \beta\right)$

For **classification**, $z = yf(x) = y\phi(x)\beta$:
(i) **0-1 Loss:** $\psi(z) = 1_{z \leq 0}$ (Also called misclassification loss)
(ii) **Exponential:** $\psi(z) = e^{-z}$, used in boosting algorithms.
(iii) **Logistic:** $\psi(z) = \log\left(1 + e^{-z}\right) / \log(2)$, used in logistic regression, and associated with a linear log-odds probabilistic model.
(iv) **Perceptron:** $\psi(z) = 1 + \max(0, -z)$
(v) **Hinge:** $\psi(z) = \max(0, 1 - z)$, used in SVM, leading to sparse solutions.
(vi) **Squared:** $\psi(z) = (1 - z)^2$

For **binary classification**, 0-1 loss is ideal but is NP hard, hence we use *convex upper bound surrogate losses.* (e.g. $\psi(z) =$ hinge, exponential, logistic) The empirical loss is:
$\hat{R}(\beta) = \frac{1}{n} \sum_{i=1}^{n} \psi\left(y_i \beta^\top \phi\left(x_i\right)\right) \implies \frac{\partial \hat{R}}{\partial \beta} = \frac{1}{n} \sum_{i=1}^{n} \psi'\left(y_i \beta^\top \phi\left(x_i\right)\right) y_i \phi\left(x_i\right)$
$\overset{\text{Hessian}}{\implies} \frac{\partial^2 \hat{R}}{\partial \beta \partial \beta^\top} = \frac{1}{n} \sum_{i=1}^{n} \psi''\left(y_i \beta^\top \phi\left(x_i\right)\right) \phi\left(x_i\right) \phi\left(x_i\right)^\top, \because y_i^2 = 1$
$\overset{\because \psi'' \geq 0}{\implies} \alpha^T \frac{\partial^2 \hat{R}}{\partial \beta \partial \beta^\top} \alpha = \frac{1}{n} \sum_{i=1}^{n} \psi''\left(y_i \beta^\top \phi\left(x_i\right)\right) \left(\alpha^T \phi\left(x_i\right)\right)^2 \geq 0, \forall \alpha \in \mathbb{R}^D$

For **regression**:
(i) **Squared:** $\psi(y, f(x)) = (y - f(x))^2$, used in least squares regression, optimal solution is the

conditional mean $E[Y|X=x]$.
(ii) **Absolute:** $\psi(y, f(x)) = |y - f(x)|$, used in least absolute deviations regression, which is less sensitive to outliers, optimal f is the conditional median $Med(Y|X=x)$.
(iii) τ **− pinball:** $\psi(y, f(x)) = 2\max\{\tau(y - f(x)), (\tau - 1)(y - f(x))\}$, used in quantile regression, optimal solution is the $\tau-$quantile of $p(y|X=x)$.
(iv) ϵ **− insensitive loss (Vapnik loss):** $\psi(y, f(x)) = \begin{cases} 0, \text{ if } |y - f(x)| \leq \epsilon \\ 1, o/w \end{cases}$, used in support vector regression, leading to sparse solution.

2.4.2 Least Square Classifier

:

$$\begin{aligned}
\widehat{\beta} &= \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \left(1 - y_i \phi\left(x_i\right)^\top \beta\right)^2 \\
&= \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} y_i^2 \left(1 - y_i \phi\left(x_i\right)^\top \beta\right)^2 \\
&= \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \phi\left(x_i\right)^\top \beta\right)^2 \\
&\implies \widehat{\beta} = \left(\mathbf{\Phi}^\top \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^\top \mathbf{y}
\end{aligned}$$

- LSE is a poor proxy of the 0-1 loss, as it penalizes well-classified points that are far from the boundary.

2.4.3 Perceptron

The surrogate perceptron loss: $\psi(z) = 1 + \max(0, -z)$
Objective function: $J(\beta) = 1 + \frac{1}{n} \sum_{i=1}^{n} \max\left(0, -y_i \phi\left(x_i\right)^\top \beta\right)$

Algorithm 9: Perceptron

```
Initialize β^(0) and set t =0.
Initialize learning rate η > 0.
Transformation of the features: φ(x).
for t in 1,...,T do
    if y_t φ(x_t)^T β^(t) ≥ 0 then
        β^(t+1) = β^(t)
    end
    else
        β^(t+1) = β^(t) + η y_t φ(x_t)
    end
end
```

2.4.4 Logistic Regression

LR as a **Plug-in Method**:
Assume $\Pr(Y = 1 \mid X = x) = \text{sig}\left(\phi(x)^\top \beta\right) \implies \log \frac{\Pr(Y=1|X=x)}{\Pr(Y=-1|X=x)} = \phi(x)^\top \beta$.

$\implies$ Bayes classifier: $h^*(x) = \begin{cases} 1, \text{ if } f(x) = \phi(x)^T\beta \geq 0 \\ -1, o/w \end{cases}$, Plug-in classifier: $\hat{h}(x) = \begin{cases} 1, \text{ if } \phi(x)^T\hat{\beta} \geq 0 \\ -1, o/w \end{cases}$

$\implies \ell(\beta) = \sum_{i=1}^n \log \text{sig}\left(y_i\phi\left(x_i\right)^\top\beta\right) = -\sum_{i=1}^n \log\left(1 + e^{-y_i\phi(x_i)^\top\beta}\right)$

$\implies \widehat{\beta} = \underset{\beta\in\mathbb{R}^p}{\arg\max} -\sum_{i=1}^n \log\left(1 + e^{-y_i\phi(x_i)^\top\beta}\right)$

LR as **ERM** under the surrogate loss: $\psi(z) = \frac{-\log(sig(z))}{\log(2)} = \frac{\log(1+e^{-z})}{\log(2)}$ ($\times\frac{1}{\log 2}$ to ensure logistic loss is an upper bound of 0-1 loss)

$\implies \hat{\beta} = \underset{\beta\in\mathbb{R}^p}{\arg\min} \frac{1}{n\log(2)} \sum_{i=1}^n \log\left(1 + e^{-y_i\phi(x_i)^\top\beta}\right)$

Hence the **objective function** is:

$J(\beta) = \frac{1}{n}\sum_{i=1}^n \log\left(1 + e^{-y_i\phi(x_i)^\top\beta}\right) = -\frac{1}{n}\sum_{i=1}^n \log \text{sig}\left(y_i\phi\left(x_i\right)^\top\beta\right)$

Recall: $\text{sig}(-z) = 1 - \text{sig}(z)$ $\quad \frac{\partial \text{sig}(z)}{dz} = \text{sig}(z)\,\text{sig}(-z)$

$\frac{\partial \log \text{sig}(z)}{dz} = \text{sig}(-z)$ $\quad \frac{\partial^2 \log \text{sig}(z)}{dz^2} = -\,\text{sig}(z)\,\text{sig}(-z)$

$\implies \nabla_\beta J(\beta) = -\frac{1}{n}\sum_{i=1}^n y_i\phi\left(x_i\right) \text{sig}\left(-y_i\phi\left(x_i\right)^\top\beta\right)$

$\implies \nabla^2_\beta J(\beta) = \frac{1}{n}\sum_{i=1}^n \text{sig}\left(y_i\phi\left(x_i\right)^\top\beta\right) \text{sig}\left(-y_i\phi\left(x_i\right)^\top\beta\right) \phi\left(x_i\right)\phi\left(x_i\right)^\top \succeq 0$

Algorithm 10: Logistic Regression (Iterative Reweighted Least Squares)

Initialize $\beta^{(0)}$ and set t =0.
Initialize learning rate $\eta > 0$, mini-batch size n_b.
Transformation of the features: $\Phi(x) = [\phi(x_1), \cdots, \phi(x_n)]$.
for $t = 1, ..., T$ **do**

Let $\mu = [\mu_1, ..., \mu_n]$ s.t. $\mu_i = sig(\phi(x_i)^T \beta^{(t)})$ and $c_i = 1_{y_i=+1}$;
Compute:

$$\nabla_{\beta^{(t)}} J(\beta^{(t)}) = -\frac{1}{n} \sum_{i=1}^{n} y_i \phi(x_i) \operatorname{sig}\left(-y_i \phi(x_i)^\top \beta^{(t)}\right) = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)(\mu_i - c_i) = \Phi^T(\mu - c)$$

$$\nabla^2_{\beta^{(t)}} J(\beta^{(t)}) = \frac{1}{n} \sum_{i=1}^{n} \operatorname{sig}\left(y_i \phi(x_i)^\top \beta^{(t)}\right) \operatorname{sig}\left(-y_i \phi(x_i)^\top \beta^{(t)}\right) \phi(x_i) \phi(x_i)^\top = \Phi^T S \Phi \succeq 0$$

Update Step:
GD: $\beta^{(t+1)} = \beta^{(t)} + \frac{\eta}{n} \sum_{i=1}^{n} y_i \phi(x_i) \operatorname{sig}\left(-y_i \phi(x_i)^\top \beta^{(t)}\right)$
SGD: for a randomly sampled batch,
$\beta^{(t+1)} = \beta^{(t)} + \frac{\eta_t}{n_b} \sum_{i=1}^{n_b} \widetilde{y}_i^{(t)} \phi\left(\widetilde{x}_i^{(t)}\right) \operatorname{sig}\left(-\widetilde{y}_i^{(t)} \phi\left(\widetilde{x}_i^{(t)}\right)^\top \beta^{(t)}\right)$
Newton-Raphson (IRLS):

$$\begin{aligned}
\beta^{(t+1)} &= \beta^{(t)} - \left(\nabla^2_\beta J\left(\beta^{(t)}\right)\right)^{-1} \nabla_\beta J\left(\beta^{(t)}\right) \\
&= \beta^{(t)} + \left(\mathbf{\Phi}^\top S^{(t)} \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^\top \left(c - \mu^{(t)}\right) \\
&= \left(\mathbf{\Phi}^\top S^{(t)} \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^\top S^{(t)} \left(\mathbf{\Phi}\beta^{(t)} + \left(S^{(t)}\right)^{-1} \left(c - \mu^{(t)}\right)\right) \\
&= \left(\mathbf{\Phi}^\top S^{(t)} \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^\top S^{(t)} z^{(t)}
\end{aligned}$$

end

- $\beta^{(t+1)}$ solves the weighted least square problem:

$$\begin{aligned}
\beta^{(t+1)} &= \arg\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} S_{ii}^{(t)} \left(z_i^{(t)} - \phi(x_i)^\top \beta\right)^2 \\
&= \arg\min_{\beta \in \mathbb{R}^p} \left(z^{(t)} - \mathbf{\Phi}\beta\right)^\top S^{(t)} \left(z^{(t)} - \mathbf{\Phi}\beta\right)
\end{aligned}$$

2.5 Discriminative Classifiers

2.5.1 k-Nearest Neighbors (kNN)

Algorithm 11: k-Nearest Neighbors

Initialize k ¿ 0.

Initialize p for the distance measurement: L_p norm $= \|x' - x_n\|_p = \left(\sum_j |x'_j - x_{ij}|^p\right)^{1/p}$.

Initialize weight w for neighbor votes.

Training:

for $x' \in$ *test set* **do**

  Get the set of k nearest neighbors of x' based on the distance measurement: $knn(x')$

  for *Class labels* $c \in \{1, ..., C\}$ **do**

    Count neighbor-votes: $\hat{f}_c(x) = \sum_{x' \in knn(x)} 1_{y'=c}$

  end

  Classification rule: label with majority $\hat{h}(x) = \underset{c=1,...,C}{\arg\max} \hat{f}_c(x)$

  Regression rule: average aross nearest neighbors $\hat{h}(x) = \frac{\sum_{x' \in knn(x)} y'}{k}$

end

2.5.2 Decision Tree

A decision tree gives a partition of $\mathcal{X}$ into R disjoint sets (regions) $\mathcal{P} = \{\mathcal{R}_1, \ldots, \mathcal{R}_R\}$, such that the fitted decision function is constant on each region $\mathcal{R}_j \subset \mathcal{X}, j = 1, \ldots, R$, i.e.

$$h(x) = \beta_j, \quad \forall x \in \mathcal{R}_j$$

Node Impurity: a good split has both sides being pure.
- For binary classification, the proportion of class 1 items in a node corresponding to a region $\mathcal{R}$ is given by

$$\eta_1 = \frac{\sum_i 1_{y_i=1} 1_{x_i \in \mathcal{R}}}{\sum_i 1_{x_i \in \mathcal{R}}}, \quad \eta_{1l} = \frac{\sum_i 1_{y_i=1} 1_{x_i \in \mathcal{R}_l}}{\sum_i 1_{x_i \in \mathcal{R}_l}}, \quad \eta_{1r} = \frac{\sum_i 1_{y_i=1} 1_{x_i \in \mathcal{R}_r}}{\sum_i 1_{x_i \in \mathcal{R}_r}}$$

Different measures of node impurity:
- Misclassification error: $1 - \max\{\eta_1, 1 - \eta_1\}$ (binary) or $1 - \max_k \eta_k$ (multi-class);
- Gini impurity: $2\eta_1(1 - \eta_1)$ (binary) or $\sum_{k=1}^K \eta_k(1 - \eta_k)$ (multi-class);
- Entropy: $-\eta_1 \log \eta_1 - (1 - \eta_1) \log(1 - \eta_1)$ (binary) or $-\sum_{k=1}^K \eta_k \log \eta_k$ (multi-class)

Comparision of the impurity measurements:
- Entropy vs misclassification error: entropy is more sensitive than misclassification error to changes in $\hat{p}_0$.
- Entropy vs Gini: They are extremely similar, with a slightly different interpretation (Gini measures probability of misclassification $p(1-p)$). Entropy involves computing a logarithm, which may be considered computationally expensive.
- Gini vs misclassification error: Gini is very similar to entropy and more sensitive than missclassification error to changes in $\hat{p}_0$ (slightly less so than entropy).
- Gini and entropy preferred: differentiable and produce purer nodes.

Algorithm 12: Decision Tree & Bagging & Random Forest

Initialize max depth d (or max splits l) of the tree.

Denote $\eta_k, \eta_{k,l}, \eta_{k,r}$ the proportion of class k items respectively in regions $\mathcal{R}$ (Parent), $\mathcal{R}_l$ (left leaf) and $\mathcal{R}_r$ (right leaf).

Initialize the impurity measurement $i(\cdot)$:

• Misclassification error: $1 - \max\{\eta_1, 1-\eta_1\}$ (binary) or $1 - \max_k \eta_k$ (multi-class);

• Gini impurity: $2\eta_1(1-\eta_1)$ (binary) or $\sum_{k=1}^{K} \eta_k(1-\eta_k)$ (multi-class);

• Entropy: $-\eta_1 \log \eta_1 - (1-\eta_1)\log(1-\eta_1)$ (binary) or $-\sum_{k=1}^{K} \eta_k \log \eta_k$ (multi-class)

Training:

for *each feature* $j = 1, ..., p$ *and each possible value of feature* j: $v \in \mathbb{R}$ **do**

Split the data: $\mathcal{R}_l = \{i : x_{ij} < v\}$ and $\mathcal{R}_r = \{i : x_{ij} \geq v\}$

Estimate parameters: $\beta_l = \frac{\sum_{i \in \mathcal{R}_l} y_i}{|\mathcal{R}_l|}$ and $\beta_r = \frac{\sum_{i \in \mathcal{R}_r} y_i}{|\mathcal{R}_r|}$

Compute the quality of the split by:

• Squared loss: $\sum_{i \in \mathcal{R}_l} (y_i - \beta_l)^2 + \sum_{i \in \mathcal{R}_r} (y_i - \beta_r)^2$; or

• impurity measure: $i(\eta_k), i(\eta_{k,l}), i(\eta_{k,r})$

Compute the proportion of samples assigned to the nodes: $q_l = \frac{\#\{i | x_i \in \mathcal{R}_l\}}{\#\{i | x_i \in \mathcal{R}\}}$ and $q_r = 1 - q_l$.

Choose split (i.e. feature j and value v) such that:

• Square loss is minimized; or

• change in the impurity measure is maximized: $i(\eta_1) - q_l i(\eta_{1,l}) - q_r i(\eta_{1,r})$

end

Bagging: Initialize number of bootstrapped samples B

for $b = 1, .., B$ **do**

Bootstrap: Draw indices $(b_1, ..., b_n)$ from $1, ..., n$ with replacement, get the bootstrapped sample $(x_{b_i}, \cdots, y_{b_i})_{i=1}^{n}$

Fit a tree model on the bootstrapped sample: $\hat{h}^b(x)$, via the **Training** process.

end

Aggregation: $\hat{h}_{Bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{h}^b(x)$ (the bagged estimator)

Random Forest:

Initialize number of features $p_{\max} = \lfloor \sqrt{p} \rfloor$ to be considered for each split.

Initialize a threshold $l_{\min}$ to terminate the tree fitting when the leaf node is small enough.

while *#leaf node* $> l_{\min}$ **do**

Do **Bagging**, but in the tree model fitting stage, only make a split with the randomly selected $p_{\max}$ features.

end

2.5.3 Bootstrap Aggregation (Bagging)

Average over the bootstrapped samples and reduce the variance of predictions:

$$\mathbb{E}_D\left[\left(\hat{h}(x) - \mathbb{E}_D[\hat{h}(x)]\right)^2\right] \geq \mathbb{E}_D\left[\left(\hat{h}_{Bag}(x) - \mathbb{E}_D\left[\hat{h}_{Bag}(x)\right]\right)^2\right]$$

- $\hat{h}_{ag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{h}^b(x) \to \bar{h}(x) = \mathbb{E}_D[\hat{h}(x)]$ a.s. as $B \to \infty$

- The conditional risk under the squared loss:

$$\mathbb{E}_D\left[\left(Y-\hat{h}_{ag}(X)\right)^2 \mid X=x\right] = \mathbb{E}_D\left[(Y-\bar{h}(X))^2 \mid X=x\right] + \mathbb{E}_D\left[\left(\bar{h}(X)-\hat{h}_{ag}(X)\right)^2 \mid X=x\right] \rightarrow \mathbb{E}_D\left[(Y-\bar{h}(X))^2 \mid X=x\right] \quad \text{as } B\rightarrow\infty$$

Summary of Bagging:
Pros: reduced variance; prevent overfitting; improve accuracy.
Cons: bootstrap samples not independent (though variance reduction still applies); small increase in bias; poorer interpretability because cannot be displayed as a single tree.
Algorithm see above.

Out-of-Bag CV: For each model built with the bootstrapped sample, use the unused sample from the training as the validation set.

2.5.4 Random Forest (RF)

Variable Importance:
Summary of RF: - Advantages of random forests: fast and easy to use; Particularly good for small/medium size data sets; Requires little tuning;
- Disadvantages: Typically worse than deep learning on huge datasets; Limited Interpretability.

2.5.5 Boosting

Train simple weak classifiers $h_t(x)$ with few parameters to learn (e.g. decision stump), typically with high bias and small variance. And predict with the weighted combination of them: (Boosting prediction rule)

$$h(x) = sign(\sum_{t=1}^{T}\beta_t h_t(x))$$

Algorithm 13: AdaBoost

Initialize weights $\bar{w}_{i,t_0}=\frac{1}{n}, \forall i=1,...,n$
Training:
for $t=1,..,T$ **do**
  Train the weak classifier h_t by minimizing the weighted classification error ($R_{\bar{w}_{i,t}}$):
  $\hat{h}_t = \arg\min_{h_t\in\mathcal{H}} \sum_{i=1}^{n}\bar{w}_{i,t}1_{y_i\neq h_t(x_i)}$ (forward stagewise additive modelling).
  Compute the contribution for this classifier: $\hat{\beta}_t = \frac{1}{2}\log\frac{1-R_{\bar{w}_{i,t}}(\hat{h}_t)}{R_{\bar{w}_{i,t}}(\hat{h}_t)}$
  Update weights on training points: $\bar{w}_{i,t+1} = \bar{w}_{i,t}e^{-\widehat{\beta}_t y_i \widehat{h}_t(x_i)}$
  Normalize weights such that they sum to 1: $\bar{\bar{w}}_{i,t+1} = \frac{\bar{w}_{i,t+1}}{\sum_{i=1}^{n}\bar{w}_{i,t+1}}$
end
Prediction:
Classifier: $\widehat{h}(x) = \text{sign}\left(\sum_{t=1}^{T}\widehat{\beta}_t\widehat{h}_t(x)\right)$

Boosting Methods as a whole:
Boosting method aims to minimise:

$$\frac{1}{n}\sum_{i=1}^{n}\psi\left(-y_i\left(\sum_{t=1}^{T}\beta_t h_t\left(x_i\right)\right)\right)$$

And the ψ-Boosting does the following:
(i) Initialize $\hat{f}_0(x)=0$
(ii) At iteration $t=1,...T$, add a new weighted weak learner into the model:

$$\left(\widehat{\beta}_t,\widehat{h}_t\right)=\underset{\beta_t\in\mathbb{R},h_t\in\mathcal{H}}{\arg\min}\frac{1}{n}\sum_{i=1}^{n}\psi\left(y_i\left(\widehat{f}_{t-1}(x)+\beta_t h_t(x)\right)\right)$$
$$\widehat{f}_t(x)=\hat{f}_{t-1}(x)+\widehat{\beta}_t\widehat{h}_t(x)$$

- if minimisation is intractable, perform GD.

Surrogate Exponential Loss: $\widetilde{L}(y,h(x))=\psi(yf(x))=e^{-yf(x)}$
AdaBoost: Forward Stagewise Additive Modelling (FSAM):
AdaBoost does ERM under the surrogate exponential loss:

$$\frac{1}{n}\sum_{i=1}^{n}e^{-y_i f(x_i)}=\frac{1}{n}\sum_{i=1}^{n}e^{-y_i\left(\sum_{t=1}^{T}\beta_t h_t(x_i)\right)}$$

- Direct minimisation is impossible, hence FSAM is a feasible stagewise/greedy approach that adds a new weak learner to the model at each time step $t=1,...T$:

$$\begin{aligned}
\left(\widehat{\beta}_t,\widehat{h}_t\right)&=\underset{\beta_t\in\mathbb{R},h_t\in\mathcal{H}}{\arg\min}\frac{1}{n}\sum_{i=1}^{n}\exp\left(-y_i\left(\widehat{f}_{t-1}(x)+\beta_t h_t(x)\right)\right)\\
&=\underset{\beta_t\in\mathbb{R},h_t\in\mathcal{H}}{\arg\min}\frac{1}{n}\sum_{i=1}^{n}w_{i,t}\exp\left(-y_i\beta_t h_t\left(x_i\right)\right),\text{ where }w_{i,t}=e^{-y_i\hat{f}_{t-1}(x_i)}\\
&=\underset{\beta_t\in\mathbb{R},h_t\in\mathcal{H}}{\arg\min}\frac{1}{n}\left[\sum_{i=1}^{n}w_{i,t}e^{\beta_t}1_{y_i\neq h_t(x_i)}+\sum_{i=1}^{n}w_{i,t}e^{-\beta_t}1_{y_i=h_t(x_i)}\right]\\
&=\underset{\beta_t\in\mathbb{R},h_t\in\mathcal{H}}{\arg\min}\frac{1}{n}\left[\left(e^{\beta_t}-e^{-\beta_t}\right)\sum_{i=1}^{n}w_{i,t}1_{y_i\neq h_t(x_i)}+e^{-\beta_t}\sum_{i=1}^{n}w_{i,t}\right]\\
&=\underset{\beta_t\in\mathbb{R},h_t\in\mathcal{H}}{\arg\min}\frac{1}{n}\left[\left(\sum_{i=1}^{n}w_{i,t}\right)\left(\left(e^{\beta_t}-e^{-\beta_t}\right)\sum_{i=1}^{n}\bar{w}_{i,t}1_{y_i\neq h_t(x_i)}+e^{-\beta_t}\right)\right]\\
\implies\hat{h}_t&=\underset{h_t\in\mathcal{H}}{\arg\min}\sum_{i=1}^{n}\bar{w}_{i,t}1_{y_i\neq h_t(x_i)}\\
\implies\hat{\beta}_t&=\arg\min\left(e^{\beta_t}-e^{-\beta_t}\right)\widehat{\epsilon}_t+e^{-\beta_t}\underset{\partial\beta_t}{\implies}\hat{\beta}_t=\frac{1}{2}\log\frac{1-\widehat{\epsilon}_t}{\widehat{\epsilon}_t},\text{ where }\widehat{\epsilon}_t=\sum_{i=1}^{n}\bar{w}_{i,t}1_{y_i\neq h_t(x_i)}\\
\widehat{f}_t(x)&=\widehat{f}_{t-1}(x)+\widehat{\beta}_t\widehat{h}_t(x)
\end{aligned}$$

Other Boost:
Logit Boost: $\psi=\log(1+e^{-x})$;

L_2 Boost: square loss, fit the residuals to get the next weak leanrer.

Summary of Boosting:
- Reduce bias (vs Bagging);
- Resistant to overfitting (the testing error typically stays flat for a large number of iterations - but will eventually go up);

3 Advanced Topics

In this section, we will go over some advanced topics in ML.

3.1 Support Vector Machines (SVM)

3.1.1 Linearly Separable Case

Find a hyperplane to separate the classes. Want to choose the one that maximizes the **margin** (i.e. twice the smallest distance from each class to the separating hyperplane).

The **optimization problem**:

$$\begin{aligned}\max_{w,b}(\text{margin}) &= \max_{w,b}\|x_i - x_j\| = \max_{w,b}\left\{(x_i - x_j)^\top \frac{(x_i - x_j)}{\|x_i - x_j\|}\right\} = \max_{w,b}\left\{(x_i - x_j)^\top \frac{w}{\|w\|}\right\} \\ &= \max_{w,b}(\frac{2}{\|w\|}), \text{ subtract the constraints} \\ &\propto \max_{w,b}\frac{1}{\|w\|} \propto \min_{w,b}\|w\|^2\end{aligned}$$

$$\text{subject to } \begin{cases}\min(w^T x_i + b) = 1, \text{ for } i : y_i = 1 \\ \max(w^T x_i + b) = -1, \text{ for } i : y_i = -1\end{cases} \quad \text{or } y_i(w^T x_i + b) \geq 1$$

3.1.2 C-SVM: Non-linearly Separable or Larger Margin case

$$\min_{w,b}\left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\mathbb{I}\left\{y_i\left(w^\top x_i + b\right) < 1\right\}\right)$$

where C controls the trade-off between maximizing margin and minimizing misclasified errors.

While 0-1 loss is computationally expensive, replace it with the hinge loss:

$$\min_{w,b}\left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\left(1 - y_i\left(w^\top x_i + b\right)\right)_+\right) = \min_{w,b}\left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i\right)$$

subject to $\xi_i \geq 0$ and $y_i\left(w^\top x_i + b\right) \geq 1 - \xi_i$.

Lagrangian: $L(w, b, \xi, \alpha, \lambda) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(w^\top x_i + b\right) - \xi_i\right) + \sum_{i=1}^{n}\lambda_i\left(-\xi_i\right)$ with $\alpha_i, \lambda_i \geq 0$.
Differentiate wrt w, b, ξ:

$$\begin{aligned}\frac{\partial L}{\partial w} &= w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0 \implies w = \sum_{i=1}^{n}\alpha_i y_i x_i \\ \frac{\partial L}{\partial b} &= \sum_i y_i \alpha_i = 0 \\ \frac{\partial L}{\partial \xi_i} &= C - \alpha_i - \lambda_i = 0 \implies \alpha_i = C - \lambda_i \overset{\because \lambda_i \geq 0}{\implies} \alpha_i \leq C\end{aligned}$$

Substitute back to Lagrangian gives the **Dual** problem: $\min_\alpha g(\alpha)$ subject to $0 \le \alpha_i \le C$ and $\sum_{i=1}^n y_i\alpha_i = 0$, where

$$\begin{aligned} g(\alpha) = &\frac{1}{2}\|w\|^2 + C\sum_{i=1}^n \xi_i + \sum_{i=1}^n \alpha_i \left(1 - y_i\left(w^\top x_i + b\right) - \xi_i\right) + \sum_{i=1}^n \lambda_i\left(-\xi_i\right) \\ =&\frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j y_i y_j x_i^\top x_j + C\sum_{i=1}^m \xi_i - \sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j y_i y_j x_i^\top x_j - b\underbrace{\sum_{i=1}^m \alpha_i y_i}_{0} \\ &+ \sum_{i=1}^m \alpha_i - \sum_{i=1}^m \alpha_i\xi_i - \sum_{i=1}^m \left(C - \alpha_i\right)\xi_i \\ =&\sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j y_i y_j x_i^\top x_j. \end{aligned}$$

Support Vectors (SVs): By condition $\alpha_i = C - \lambda_i$ and by complementary slackness,

(i) **Non-margin SV**: $\alpha_i = C > 0 \implies \begin{cases} 1 - \xi_i = y_i(w^T x_i + b) \\ \lambda_i = 0 \implies \xi_i \ge 0 \end{cases}$

(ii) **Margin SV**: $0 < \alpha_i < C \implies \begin{cases} 1 - \xi_i = y_i(w^T x_i + b) \\ \lambda_i > 0 \implies \xi_i = 0 \end{cases}$

(iii) **Non-SV**: $\alpha_i = 0 \implies \begin{cases} 1 - \xi_i \le y_i(w^T x_i + b) \\ \lambda_i > 0 \implies \xi_i = 0 \end{cases}$

3.1.3 $\nu-$SVM

$$\min_{w,\rho,\xi}\left(\frac{1}{2}\|w\|^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^n \xi_i\right)$$

subject to $\rho \ge 0, \xi_i \ge 0, y_i(w^T x_i + b) \ge 1 - \xi_i$

Now the margin width is $2\rho/\|w\|$, hence ν controls the strength of the pressure to maximize the margin width, with larger values of ν encouraging larger margins (typically at the cost of more margin errors).

Lagrangian: for $\alpha_i, \beta_i, \gamma_i \ge 0$

$$L(w, b, \xi, \rho, \alpha, \beta, \gamma) = \frac{1}{2}\|w\|_{\mathcal{H}}^2 + \frac{1}{n}\sum_{i=1}^n \xi_i - \nu\rho + \sum_{i=1}^n \alpha_i\left(\rho - y_i\left(w^\top x_i + b\right) - \xi_i\right) + \sum_{i=1}^n \beta_i\left(-\xi_i\right) + \gamma(-\rho)$$

Differentiate wrt w, b, ξ, ρ and set to zero:

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

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

$$\alpha_i + \beta_i = \frac{1}{n} \overset{\because \beta_i \geq 0}{\Longrightarrow} 0 \leq \alpha_i \leq \frac{1}{n}$$

$$\nu = \sum_{i=1}^{n} \alpha_i - \gamma\nu \overset{\because \gamma \geq 0}{\leq} \sum_{i=1}^{n} \alpha_i$$

Substituing into Langrangian gives the **Dual** formulation: $\max_\alpha g(\alpha)$ subject to $\sum_{i=1}^{n} \alpha_i \geq \nu$ and $0 \leq \alpha_i \leq 1/n$, where:

$$\begin{aligned} g(\alpha) =& \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^\top x_j + \frac{1}{n} \sum_{i=1}^{n} \xi_i - \rho\nu - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n} \alpha_i \rho - \sum_{i=1}^{n} \alpha_i \xi_i \\ & - \sum_{i=1}^{n} \left(\frac{1}{n} - \alpha_i \right) \xi_i - \rho \left(\sum_{i=1}^{n} \alpha_i - \nu \right) \\ =& -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^\top x_j \end{aligned}$$

Significance of ν: ν corresponds to an upper bound on the proportion of margin errors and a lower bound on the proportion of the overall number of support vectors (hence tuning ν is much more interpretable than tuning C).

$$\frac{|N(\alpha)|}{n} \leq \nu \leq \frac{|N(\alpha)| + |M(\alpha)|}{n}$$

where $N(\alpha)$ is the set of non-margin support vectors (i.e. margin errors) and $M(\alpha)$ is the set of margin support vectors.

3.2 Kernel Method

Employ linear tools on a non-linearly transformed feature space.

3.2.1 Hilbert Space

Inner product:
(i) $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle_{\mathcal{H}} = \alpha_1 \langle f_1, g \rangle_{\mathcal{H}} + \alpha_2 \langle f_2, g \rangle_{\mathcal{H}}$ (linear)
(ii) $\langle f, g \rangle_{\mathcal{H}} = \langle g, f \rangle_{\mathcal{H}}$(symmetric)
(iii) $\langle f, f \rangle_{\mathcal{H}} \geq 0$
(iv) $\langle f, f \rangle_{\mathcal{H}} = 0 \iff f = 0$
Hilbert Space: Vector space on which an inner product is defined, along with an additional technical condition
Kernel: A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if $\exists \mathcal{H}$ (Hilbert space) and a map $\varphi : \mathcal{X} \to \mathcal{H}$ s.t. $\forall x, x' \in \mathcal{X}, k(x, x') := \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}$

Positive definite: A symmetric function $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is positive definite if $\forall n \geq 1, (a_1, \cdots, a_n), (x_1, \cdots, x_n) \in \mathbb{R}^{\kappa}, \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j k(x_i, x_j) \geq 0$.
• **strictly positive definite**: the equality holds only when all $\alpha_i = 0$.
• All kernel functions (defined as inner products between some features) are positive definite.
$\because \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j k(x_i, x_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} \langle a_i\varphi(x_i), a_j\varphi(x_j)\rangle_{\mathcal{H}} = \|\sum_{i=1}^{n} a_i\varphi(x_i)\|_{\mathcal{H}}^2 \geq 0$

3.2.2 Reproducing Kernel Hilbert Spaces (RKHS)

Reproducing kernel: A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a reproducing kernel if:
(i) $\forall x \in \mathcal{X}, \quad k_x := k(\cdot, x) \in \mathcal{H}$ and;
(ii) $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \langle f, k(\cdot, x)\rangle_{\mathcal{H}} = f(x)$ (the reproducing property).
- If $\mathcal{H}$ has a reproducing kernel, then it is a RKHS.

Moore-Aronszajn theorem: Every positive definite function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is also a reproducing kernel with a unique corresponding RKHS. (i.e. reproducing kernel $\iff$ a kernel as an inner product between features $\iff$ a positive definite function)
• **Derivation**: For an arbitrary positive definite function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, define
- *vector space*: $\mathcal{H}_0 := \{\sum_{i=1}^{r} \alpha_i k(\cdot, x_i)\}_{r\in\mathbb{N},\alpha_i\in\mathbb{R},x_i\in\mathcal{X}}$;
- *function*: $h : (\mathcal{X} \to \mathbb{R}) \times (\mathcal{X} \to \mathbb{R} \to \mathbb{R}$ s.t. $h(k(\cdot, x), k(\cdot, x')) := k(x, x')$;
- $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ operation: for any 2 functions $f(\cdot) = \sum_{i=1}^{r} \alpha_i k(\cdot, x_i)$ and $g(\cdot) = \sum_{j=1}^{s} \alpha_i k(\cdot, x'_j)$, $\langle f, g\rangle_{\mathcal{H}} := \sum_{i=1}^{r}\sum_{j=1}^{s} \alpha_i\beta_j h\left(k(\cdot, x_i), k\left(\cdot, x'_j\right)\right) = \sum_{i=1}^{r}\sum_{j=1}^{s} \alpha_i\beta_j k\left(x_i, x'_j\right)$
Then, show $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ is a valid *inner product*:
(i) $\langle af_1 + bf_2, g\rangle_{\mathcal{H}} = \sum_{i=1}^{r_1+r_2}\sum_{j=1}^{s} \alpha_i\beta_j k\left(x_i, x'_j\right) = \sum_{j=1}^{s} \beta_j \left(a\sum_{i=1}^{r_1} \alpha_{i,1} k\left(x_{i,1}, x'_j\right) + b\sum_{i=1}^{r_2} \alpha_{i,2} k\left(x_{i,2}, x'_j\right)\right)$
$= a\sum_{i=1}^{r_1}\sum_{j=1}^{s} \alpha_{i,1}\beta_j k\left(x_{i,1}, x'_j\right) + b\sum_{i=1}^{r_2}\sum_{j=1}^{s} \alpha_{i,2}\beta_j k\left(x_{i,2}, x'_j\right) = a\langle f_1, g\rangle_{\mathcal{H}} + b\langle f_2, g\rangle_{\mathcal{H}}$;
(ii) Symmetry condition is trivial;
(iii) $\langle f, f\rangle_{\mathcal{H}} = \sum_{i=1}^{r}\sum_{j=1}^{r} \alpha_i\alpha_j k(x_i, x_j) \geq 0 \because k(x, x')$ is positive definite (by def);
(iv) $\|f\|_{\mathcal{H}_k}^2 = \langle f, f\rangle_{\mathcal{H}} = \sum_{i=1}^{r} \alpha_i \sum_{j=1}^{r} \alpha_j k(x_i, x_j) = \sum_{i=1}^{r} \alpha_i f(x_i) = 0$ if $f(x) = 0$ proof for reverse?
Finally, show reproducing property:
$\langle f, k(\cdot, x)\rangle_{\mathcal{H}} = \langle\sum_{i=1}^{r} \alpha_i k(\cdot, x_i), k(\cdot, x)\rangle_{\mathcal{H}} = \sum_{i=1}^{r} \alpha_i \langle k(\cdot, x_i), k(\cdot, x)\rangle_{\mathcal{H}} = \sum_{i=1}^{r} a_i k\langle x, x_i\rangle_{\mathcal{H}} = f(x)$.

• Any RKHS has a unique reproducing kernel, because:
$\langle f, k_1(\cdot, x) - k_2(\cdot, x)\rangle_{\mathcal{H}} = \langle f, k_1(\cdot, x)\rangle_{\mathcal{H}} - \langle f, k_2(\cdot, x)\rangle_{\mathcal{H}} = f(x) - f(x) = 0, \forall f \in \mathcal{H}, x \in \mathcal{X}$
- Any reproducing kernel (or any positive definite function) has a unique RKHS.

3.2.3 Kernel Operations

• **Sum** of kernels are kernels, but differences of kernels may not: k_1, k_2 are kernels on $\mathcal{X} \implies \alpha_1 k_1 + \alpha_2 k_2$ is a kernel on $\mathcal{X}$, given $\alpha_1, \alpha_2 > 0$.
• Kernel properties preserves over **mappings between spaces**: If $A : \S \to \tilde{\mathcal{X}}$ and k a kernel on $\tilde{\mathcal{X}}$, then $k(A(x), A(x'))$ a kernel on $\tilde{\mathcal{X}}$.
• **Products** of kernels are kernels: $\kappa((x, y), (x', y')) = k(x, x')l(y, y')$ is a kernel on $\mathcal{X} \times \mathcal{Y}$, given k on $\mathcal{X}$ and l on $\mathcal{Y}$. (If $\mathcal{X} = \mathcal{Y}$, then κ is a kernel on $\mathcal{X}$)
proof: $k(x, x')\, l(y, y') = \varphi(x)^{\top}\varphi(x')\,\psi(y')^{\top}\psi(y) = \mathrm{tr}\left(\psi(y)\varphi(x)^{\top}\varphi(x')\,\psi(y')^{\top}\right) = \left\langle\varphi(x)\psi(y)^{\top}, \varphi(x')\,\psi(y')^{\top}\right\rangle$, given by the property of inner product of 2 matrices: $\langle A, B\rangle = tr(A^T B)$ for $A \in \mathbb{R}^{M\times N}$ and $B \in \mathbb{R}^{N\times M}$.

3.2.4 Various types of kernels

Combined with the operations together:
(i) **Polynomial kernel**: $k(x,x') := (\langle x,x'\rangle + c)^m$ is a valid kernel, for $x,x' \in \mathbb{R}^p, p \geq 1, m \geq 1, m \in \mathbb{Z}, c \geq 0$. (To prove, expand the polynomial as a sum of kernels of different powers.)
(ii) **Exponential kernel** $\exp(\langle x,x'\rangle)$ is a valid kernel by Taylor series expansion.
(iii) **Gaussian RBF kernel** (or *squared exponential kernel / exponentiated quadratic kernel*): $k(x,x') := \exp\left(-\frac{1}{2\gamma^2}\|x-x'\|^2\right)$, where γ is the **length scale**.
• RKHS of RBF kernel is infinitely dimensional, and all function in the RKHS is infinitely differentiable, hence smooth.
(iv) **Matérn kernel** $k(x,x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}}{\gamma}\|x-x'\|\right)^\nu K_\nu\left(\frac{\sqrt{2\nu}}{\gamma}\|x-x'\|\right), \quad \nu > 0, \gamma > 0$, where K_ν is the modified Bessel function of the second kind of order ν.
• If $\nu = s + 1/2$ for non-negative integers s, then Matérn kernel takes a simpler form, e.g.:
- $\nu = 1/2 : k(x,x') = \exp\left(-\frac{1}{\gamma}\|x-x'\|\right)$
- $\nu = 3/2 : k(x,x') = \left(1+\frac{\sqrt{3}}{\gamma}\|x-x'\|\right)\exp\left(-\frac{\sqrt{3}}{\gamma}\|x-x'\|\right)$
- $\nu = 5/2 : k(x,x') = \left(1+\frac{\sqrt{5}}{\gamma}\|x-x'\|+\frac{5}{3\gamma^2}\|x-x'\|^2\right)\exp\left(-\frac{\sqrt{5}}{\gamma}\|x-x'\|\right)$,
and its RKHS consists of s times differentiable functions with square integrable derivatives of order up to s + 1, its RKHS norms also directly penalize the derivatives of f, e.g. for $\nu = 3/2$:
$\|f\|^2_{\mathcal{H}_k} \propto \int f''(x)^2 dx + \frac{6}{\gamma^2}\int f'(x)^2 dx + \frac{9}{\gamma^4}\int f(x)^2 dx$
• $\nu \to \infty \implies$ Matérn $\to$ RBF.
(v) **Rational quadratic kernel**: a scale mixture of Gaussian kernels, for $k_\theta(x,x') = \exp\left(-\theta\|x-x'\|^2\right)$ and gamma density $p(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{\alpha-1}\exp(-\beta\theta)$,
$k(x,x') = \int_0^\infty k_\theta(x,x')\,p(\theta)d\theta = \frac{\beta^\alpha}{\Gamma(\alpha)}\int_0^\infty \exp\left(-\theta\left(\|x-x'\|^2+\beta\right)\right)\theta^{\alpha-1}d\theta = \frac{\beta^\alpha}{\Gamma(\alpha)}\frac{\Gamma(\alpha)}{\left(\|x-x'\|^2+\beta\right)^\alpha} = \left(1+\frac{\|x-x'\|^2}{\beta}\right)^{-\alpha}$
•$\beta = 2\alpha\gamma^2$ and $\alpha \to \infty \implies$ Rational quadratic $\to$ Gaussian RBF, hence Rational quadratic RKHSs contain functions which vary smoothly across multiple length-scales (γ).

3.2.5 Representer Theorem

There is always a solution to the optimization problem:

$$\text{argmin}_{f\in\mathcal{H}_k}\hat{R}(f) + \Omega\left(\|f\|^2_{\mathcal{H}_k}\right)$$

that takes the form: $f^* = \sum_{i=1}^n \alpha_i k(\cdot,x_i), \quad \alpha_i \in \mathcal{R}$
• If Ω is strictly increasing, then all solutions take this form.

Proof: Suppose f is a solution. Denote $f = f_s + f_\perp$, where $f_s = \sum_{i=1}^n \alpha_i k(\cdot,x_i)$ is the projection of f onto $span\{k(\cdot,x_i) : i = 1,...,n\}$ and $f_\perp$ is orthogonal to the span.
Then, $\|f\|^2_{\mathcal{H}_k} = \|f_s\|^2_{\mathcal{H}_k} + \|f_\perp\|^2_{\mathcal{H}_k} \geq \|f_s\|^2_{\mathcal{H}_k} \implies \Omega\left(\|f\|^2_{\mathcal{H}_k}\right) \geq \Omega\left(\|f_s\|^2_{\mathcal{H}_k}\right)$.
Also, each individual term of f and f_s is equal, hence so do the empirical risk: $f(x_i) = \langle f, k(\cdot,x_i)\rangle_{\mathcal{H}_k} = \langle f_s + f_\perp, k(\cdot,x_i)\rangle_{\mathcal{H}_k} = \langle f_s, k(\cdot,x_i)\rangle_{\mathcal{H}_k} = f_s(x_i) \implies L(y_i, f(x_i), x_i) = L(y_i, f_s(x_i), x_i) \implies \hat{R}(f) = \hat{R}(f_s)$.

So, f_s is also a solution.
If Ω is strictly increasing, then must have $\|f_\perp\|_{\mathcal{H}_k} = 0$, hence $f_\perp = 0$ and $f = f_s$ is the only solution.

3.2.6 Kernel SVM

Recall the hinge loss SVM objective function in RKHS, with b dropped for simplicity:

$$\min_{w\in\mathcal{H}} \left(\frac{1}{2}\|w\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n} \left(1 - y_i \left\langle w, k\left(x_i, \cdot\right)\right\rangle_{\mathcal{H}}\right)_+ \right)$$

By Representer Thm, $w = \sum_{i=1}^n \beta_i k(x_i, \cdot)$
Substituting w and introducing ξ:
$\min_{\beta,\xi} \left(\frac{1}{2}\beta^\top K\beta + C\sum_{i=1}^n \xi_i\right)$ subject to $\xi_i \geq 0$ and $y_i \sum_{j=1}^n \beta_j k(x_i, x_j) \geq 1 - \xi_i$, where $K_{ij} = k(x_i, x_j)$.
Following a similar calculation as in the SVM section, the dual takes the same form as before: $g(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j k\left(x_i, x_j\right)$ subject to $0 \leq \alpha_i \leq C$ and $w = \sum_{i=1}^n y_i\alpha_i k(x_i, \cdot)$ ($\because \beta_i = y_i\alpha_i$ by differentiating the Lagrangian wrt β_i and setting to zero).

3.2.7 Kernel PCA

Assumptions:
- finite-dimensional feature space $\mathcal{H} = \mathbb{R}^M$;
- features are **centred**: $\frac{1}{n}\varphi(x_i) = 0$. (Any kernel matrix can be tranformed to be centred, proof?)

Then the $M \times M$ sample covariance is: $\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^n \varphi\left(x_i\right)\varphi\left(x_i\right)^\top = \frac{1}{n-1}\Phi^\top\Phi$, where $\Phi \in \mathbb{R}^{n\times M}$ is the feature representation.
Q: Want to solve the eigenvalue problem $Sv_m = \lambda_m v_m, m = 1, ..., M$
Note that whenever $\lambda_m > 0$, the eigenvectors v_m lie in the linear span of feature vectors span $\{\varphi(x_i) : i = 1, ..., n\}$, that is:
$v_m = \sum_{i=1}^n a_{mi}\varphi(x_i) \iff \lambda_m v_m = \mathbf{S}v_m = \frac{1}{n-1}\sum_{i=1}^n \varphi\left(x_i\right)\left(\varphi\left(x_i\right)^\top v_m\right)$ $(*)$, where $a_{mi} = \frac{1}{\lambda_m(n-1)}\left(\varphi\left(x_i\right)^\top v_m\right)$
Substitute v_m back to $(*)$ gives: $\mathbf{S}v_m = \frac{1}{n-1}\sum_{i=1}^n \varphi\left(x_i\right)\sum_{\ell=1}^n a_{m\ell}k\left(x_i, x_\ell\right) = \lambda_m\sum_{i=1}^n a_{mi}\varphi\left(x_i\right)$
Multiply both sides by $\varphi(x_j), j = 1, ..., n$: $\frac{1}{n-1}\sum_{i=1}^n k\left(x_j, x_i\right)\sum_{\ell=1}^n a_{m\ell}k\left(x_i, x_\ell\right) = \lambda_m\sum_{i=1}^n a_{mi}k\left(x_j, x_i\right)$
$\implies \mathbf{K}^2 = \lambda_m(n-1)\mathbf{K}\alpha_m$, so α_m is the eigenvector of $\mathbf{K}$ with the eigenvalue λ_m, if $\mathbf{K}$ invertible.

Alternatively, consider Eigen-decomposition of $\mathbf{K} = UDU^T$, where u_m is the m-eigenvector of $\mathbf{K}$ with unit norm. Recall $1 = v_m^T v_m = a_m^T\mathbf{K}a_m = \lambda_m(n-1)a_m^T a_m$, to ensure v_m has unit vector, rescale $a_m = {}^{u_m}/\sqrt{\lambda(n-1)}$. So, the PC projections:
$z_i^{(m)} = v_m^\top\varphi\left(x_i\right) = \left(\sum_{j=1}^n a_{mj}\varphi\left(x_j\right)\right)^\top\varphi\left(x_i\right) = \sum_{j=1}^n a_{mj}k\left(x_j, x_i\right)$
$\implies \mathbf{z}^{(m)} = \mathbf{K}a_m = \lambda_m(n-1)a_m = \sqrt{\lambda_m(n-1)}u_m$ (No need explicit feature transformations!)

3.2.8 Representation of probabilities in RKHS

Kernel Mean Embedding: represent probability in RKHS $P \mapsto \mu_k(P) = \mathbb{E}_{X\sim P}k(\cdot, X) \in \mathcal{H}_k$ •
converts funtion $f \in \mathcal{H}_k$ to its mean: $\langle\mu_k(P), f\rangle_{\mathcal{H}_k} = \langle\mathbb{E}_{X\sim P}k(\cdot, X), f\rangle_{\mathcal{H}_k} = \mathbb{E}_{X\sim P}\langle k(\cdot, X), f\rangle_{\mathcal{H}_k} = \mathbb{E}_{X\sim P}f(X), \quad \forall f \in \mathcal{H}_k$
• exists whenever $f \to \mathbb{E}_{X\sim P}f(X)$ is bounded:

$\mathbb{E}_{X\sim P} f(X) = \mathbb{E}_{X\sim P}\langle f, k(\cdot, X)\rangle \leq \|f\|_{\mathcal{H}_k}\mathbb{E}_{X\sim P}\|k(\cdot, X)\|_{\mathcal{H}_k} \leq \sqrt{M}\|f\|_{\mathcal{H}_k}$, by Cauchy-Schwarz
• f itself is a kernel mean embedding from $X \to Y$: $\langle \mu_k(P), \mu_k(Q)\rangle_{\mathcal{H}_k} = \mathbb{E}_{X\sim P}\mathbb{E}_{Y\sim Q}k(X, Y)$

Maximum mean discrepancy (MMD): (squared) distances between probability measures in RKHS.

$$\begin{aligned} \mathrm{MMD}_k^2(P, Q) &= \|\mu_k(P) - \mu_k(Q)\|_{\mathcal{H}_k}^2 \\ &= \mathbb{E}_{X,X'\overset{iid}{\sim}dP} k\left(X, X'\right) + \mathbb{E}_{Y,Y'\overset{iid}{\sim}Q} k\left(Y, Y'\right) - 2\mathbb{E}_{X\sim P, Y\sim Q}k(X, Y) \\ \iff \mathrm{MMD}_k(P, Q) &= \sup_{f\in\mathcal{H}_k:\|f\|_{\mathcal{H}_k}\leq 1} |\mathbb{E}_{X\sim P}f(X) - \mathbb{E}_{Y\sim Q}f(Y)| \end{aligned}$$

(proof?)
• **witness function**: $\mu_k(P) - \mu_k(Q)$;
• **characteristic**: kernels s.t. $MMD_k(P, Q) = 0 \implies P = Q$. (e.g. Gaussian, Matern family and rational quadratic.)
• An estimator: $\widehat{\mathrm{MMD}}_k^2(P, Q) = \frac{1}{n_x(n_x-1)}\sum_{i\neq j} k\left(x_i, x_j\right) + \frac{1}{n_y(n_y-1)}\sum_{i\neq j} k\left(y_i, y_j\right) - \frac{2}{n_x n_y}\sum_{i=1}^{n_x}\sum_{j=1}^{n_y} k\left(x_i, y_j\right)$ (i.e. the difference between within-sample average similarity (self- similarity excluded) and the between-sample average similarity.)

Hilbert-Schmidt independence criterion (HSIC): For $X \in \mathcal{X}$, $Y \in \mathcal{Y}$, $k_{\mathcal{X}}/k_{\mathcal{Y}}$ kernels on $\mathcal{X}/\mathcal{Y}$, HSIC of X, Y is the squared MMD between the joint measure P_{XY} and the product of the marginals $P_X P_Y$, computed with the *product kernels* $k = k_{\mathcal{X}} \otimes k_{\mathcal{Y}}$:

$$\begin{aligned} \Xi_{k_{\mathcal{X}}, k_{\mathcal{Y}}}(X,Y) &= \|\mu_k\left(P_{XY}\right) - \mu_k\left(P_X P_Y\right)\|_{\mathcal{H}_k}^2 \\ &= \|\mathbb{E}_{XY}\left[k_{\mathcal{X}}(., X) \otimes k_{\mathcal{Y}}(., Y)\right] - \mathbb{E}_X k_{\mathcal{X}}(., X) \otimes \mathbb{E}_Y k_{\mathcal{Y}}(., Y)\|_{\mathcal{H}_k}^2 \end{aligned}$$

where $\varphi(x, y) = k_{\mathcal{X}}(\cdot, x) \otimes k_{\mathcal{Y}}(\cdot, y)$ s.t. $(\varphi(x, y))(x', y') = k_{\mathcal{X}}(x', x)k_{\mathcal{Y}}(y', y)$
- measures dependence between random variables taking values in some generic domains (e.g. random vectors, strings, or graphs)
• $\Xi_{k_{\mathcal{X}}k_{\mathcal{Y}}}(X, Y) = 0 \implies$ independence;
• Bounded $k_{\mathcal{X}}, k_{\mathcal{Y}}$ is the sufficient condition for HSIC to be defined.
• An estimator: $\widehat{\Xi_{k_{\mathcal{X}},k_{\mathcal{Y}}}}(X, Y) = \frac{1}{m^2}\mathrm{tr}(\widetilde{K}\widetilde{L}) = \frac{1}{m^2}\sum_{i=1}^m\sum_{j=1}^m \widetilde{K}_{ij}\widetilde{L}_{ij}$, where $\widetilde{K} = HKH$ ($K_{ij} = k(x_i, x_j)$) and $\widetilde{L} = HLH$ ($L_{ij} = k(y_i, y_j)$), $H = I - \frac{1}{m}1$

Detials omited, see Section 4.7 of Advanced Topics in SML notes.

3.3 Bayesian Machine Learning (BML)

Bayes Rule: $p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_\Theta p(\mathcal{D}|\theta)p(\theta)d\theta} = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$
• **Update distribution** for new data: $p\left(\theta \mid \mathcal{D}_1, \mathcal{D}_2\right) = \frac{p(\mathcal{D}_2|\theta,\mathcal{D}_1)p(\theta|\mathcal{D}_1)}{p(\mathcal{D}_2|\mathcal{D}_1)} = \frac{p(\mathcal{D}_2|\theta,\mathcal{D}_1)p(\mathcal{D}_1|\theta)p(\theta)}{p(\mathcal{D}_2|\mathcal{D}_1)p(\mathcal{D}_1)}$

Everything we need about the posterior:
• Posterior mode: $\widehat{\theta}^{\mathrm{MAP}} = \arg\max_{\theta\in\Theta} p(\theta \mid \mathcal{D})$ (maximum a posteriori)
• Posterior mean: $\widehat{\theta}^{\mathrm{mean}} = \mathbb{E}[\theta \mid \mathcal{D}]$
• Posterior variance: $\mathrm{Var}[\theta \mid \mathcal{D}]$

• Posterior expectations of functions of parameters: $\mathbb{E}[g(\theta) \mid \mathcal{D}]$ for some $g : \Theta \to \mathbb{R}^s$

Posterior predictive distribution: $p(\mathcal{D}^* \mid \mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}[p(\mathcal{D}^* \mid \theta)] = \int p(\mathcal{D}^* \mid \theta)\, p(\theta \mid \mathcal{D}) d\theta$
• In supervised learning: $p(y \mid x, \mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}[p(y \mid x, \theta)] = \int p(y \mid x, \theta) p(\theta \mid \mathcal{D}) d\theta$

Bayesian Naive Bayes:
The simple NB: $p(y_i = k \mid \theta) = \pi_k, \quad p(x_i \mid y_i = k, \theta) = \prod_{j=1}^p \phi_{kj}^{(j)} (1 - \phi_{kj})^{1 - x_i^{(j)}}$
$\implies MLE : \hat{\pi}_k = \frac{n_k}{n}, \quad \hat{\phi}_{kj} = \frac{\sum_{i:y_i=k} x_i^{(j)}}{n_k} = \frac{n_{kj}}{n_k}$, where $n_k = \sum_{i=1}^n \mathbb{I}\{y_i = k\}, n_{kj} = \sum_{i=1}^n \mathbb{I}\left(y_i = k, x_i^{(j)} = 1\right)$
But this is problematic for extreme values, e.g. if $n_{kl} = 0$, then $\hat{\phi}_{kl} = 0$, hence:
$p(\tilde{y} = k \mid \tilde{x}$ with ℓ-th entry equal to $1, \hat{\theta}) \propto \hat{\pi}_k \prod_{j=1}^p \left(\hat{\phi}_{kj}\right)^{\tilde{x}^{(j)}} \left(1 - \hat{\phi}_{kj}\right)^{1 - \tilde{x}^{(j)}} = 0.$
Alternatively, we can use the conjugate prior $Dir\left((\alpha_k)_{k=1}^K\right)$ for π_k and $Beta(a, b)$ for ϕ_{kj} of the same likelihood:

$$\begin{aligned} p(\mathcal{D} \mid \theta) = \prod_{i=1}^n p(x_i, y_i \mid \theta) &= \prod_{i=1}^n \prod_{k=1}^K \left(\pi_k \prod_{j=1}^p \phi_{kj}^{x_i^{(j)}} (1 - \phi_{kj})^{1 - x_i^{(j)}} \right)^{\mathbb{I}(y_i = k)} \\ &= \prod_{k=1}^K \pi_k^{n_k} \prod_{j=1}^p \phi_{kj}^{n_{kj}} (1 - \phi_{kj})^{n_k - n_{kj}} \end{aligned}$$

Since likelihood factorizes, the posterior also factorizes and posterior for π_k is $Dir\left((\alpha_k + n_k)_{k=1}^K\right)$, and for ϕ_{kj} is $Beta(a + n_k, b + n_k - n_{kj})$. Hence the predictive distribution is:

$$\begin{aligned} p(\tilde{y} = k \mid \tilde{x}, \mathcal{D}) &= \frac{p(\tilde{y} = k \mid \mathcal{D}) p(\tilde{x} \mid \tilde{y} = k, \mathcal{D})}{p(\tilde{x} \mid \mathcal{D})} \\ &\propto \frac{\alpha_k + n_k}{\sum_{l=1}^K \alpha_l + n} \prod_{j=1}^p \left(\frac{a + n_{kj}}{a + b + n_k} \right)^{\tilde{x}^{(j)}} \left(\frac{b + n_k - n_{kj}}{a + b + n_k} \right)^{1 - \tilde{x}^{(j)}} \end{aligned}$$

with $p(\tilde{y} = k \mid \mathcal{D}) = \frac{\alpha_k + n_k}{\sum_{l=1}^K \alpha_l + n}, \quad p\left(\tilde{x}^{(j)} = 1 \mid \tilde{y} = k, \mathcal{D}\right) = \frac{a + n_{kj}}{a + b + n_k}$.

Bayesian Decision Theory Choose the prediction that minimizes the **posterior expected loss**:

$$\begin{aligned} f(x) &:= \arg\min_{\hat{y}} \mathbb{E}_{Y \sim p(y|x,\mathcal{D})}[L(Y, \hat{y}, x)] \\ &= \arg\min_{\hat{y}} \iint L(y, \hat{y}, x) p(y \mid x, \theta) p(\theta \mid \mathcal{D}) dy d\theta \end{aligned}$$

Make a decision: $d^* := \operatorname{argmin}_d \mathbb{E}_{\hat{\theta} \sim p(\theta|\mathcal{D})}[L(d, \hat{\theta})]$
• Model parameters θ does not depend on the loss, and the loss only influence how we make decision.
• Pros:
- can ascertain and critique the model's fit to data prior to making decisions;
- can quantify the model's uncertainties;
- can propagate all predictive information and uncertainty through the posterior predictive rather than being forced to return point estimates;
- can use the same posterior distribution to derive multiple different decisions or predictions.
• Cons:

- significantly more computationally expensive and less scalable than ERM;
- different approximations emphasise differ aspects of the posterior, and the choice of approximations should reflect the aspects of the posterior that are important for the loss function, which breaks the separation between modelling/inference and decision making.

A Fundamental Assumption: Data points are conditionally independent given the parameter values, i.e. $p(\mathcal{D} \mid \theta) = \prod_{n=1}^{N} p(x_n \mid \theta)$.

Bayesian Model Selection: **Bayes Factor** $= \frac{P(\mathcal{D}|\mathcal{M})}{P(\mathcal{D}|\mathcal{M}')} = \frac{P(\mathcal{D}|\theta_{\mathcal{M}},\mathcal{M})P(\theta_{\mathcal{M}}|\mathcal{M})}{P(\mathcal{D}|\theta_{\mathcal{M}'},\mathcal{M}')P(\theta_{\mathcal{M}'}|\mathcal{M}')}$
• Model evidence $P(\mathcal{D}|\mathcal{M})$ is the prob- ability that a set of randomly selected parameter values (under the prior) inside the model would generate dataset $\mathcal{D}$. (Hence too simple models are less likely to generate the dataset, whereas too complex models generates too many other things and are also less likely to generate the dataset.)

3.3.1 Approximate Bayesian Inference

Prominent Classes of Approximate Bayesian Inference Methods: Markov chain Monte carlo (MCMC), Importance Sampling, Sequantial MC, Approximate Bayesian computation (ABC), Laplace Approximation, Variational inference.

Laplace Approximation (or saddle-point approximation): approximate the posterior distribution $p(\theta|\mathcal{D})$ with a (multivariate) Gaussian distribution.
Suppose $\hat{\theta}^{MAP}$ exists, the Taylor series expansion of the posterior is:

$$\begin{aligned}
\log p(\theta \mid \mathcal{D}) &= \log p\left(\widehat{\theta}^{\mathrm{MAP}} \mid \mathcal{D}\right) + \underbrace{\frac{\partial \log p(\theta \mid \mathcal{D})}{\partial \theta}_{\theta=\widehat{\theta}^{\mathrm{MAP}}}}_{=0}\left(\theta - \widehat{\theta}^{\mathrm{MAP}}\right) \\
&\quad + \frac{\partial^2 \log p(\theta \mid \mathcal{D})}{\partial \theta^2}\bigg|_{\theta=\widehat{\theta}^{\mathrm{MAP}}} \frac{\left(\theta - \widehat{\theta}^{\mathrm{MAP}}\right)^2}{2} + \mathcal{O}\left(\left(\theta - \widehat{\theta}^{\mathrm{MAP}}\right)^3\right) \\
&\approx \log p\left(\widehat{\theta}^{\mathrm{MAP}} \mid \mathcal{D}\right) - \frac{\tau}{2}\left(\theta - \widehat{\theta}^{\mathrm{MAP}}\right)^2, \text{ where } \tau = -\frac{\partial^2 \log p(\theta \mid \mathcal{D})}{\partial \theta^2} \geq 0 \\
&\sim \log \mathcal{N}\left(\widehat{\theta}^{\mathrm{MAP}}, \left(-\frac{\partial^2 \log p(\theta \mid \mathcal{D})}{\partial \theta^2}\right)^{-1}\right)
\end{aligned}$$

• Also works similarly in higher dimensions: $p(\theta|\mathcal{D}) \sim N(\hat{\theta}^{MAP}, \Sigma)$, where,
$\Sigma^{-1} = -\frac{\partial^2 \log p(\theta|\mathcal{D})}{\partial\theta\partial\theta^T}\,|_{\theta=\hat{\theta}^{MAP}} = \frac{\partial^2 \log J(\theta)}{\partial\theta\partial\theta^T}_{\theta=\hat{\theta}^{MAP}}$ ($J(\theta) = -\log p(\theta, \mathcal{D})$ is the *energy function*. (We can do this because $\log p(\theta|\mathcal{D})$ and $\log p(\theta, \mathcal{D})$ agree up to a constant.)

Importance Sampling:

$$\begin{aligned}
\mathbb{E}_{p(\theta|\mathcal{D})}[f(\theta)] &= \int f(\theta) p(\theta \mid \mathcal{D}) d\theta = \int f(\theta) \frac{p(\theta \mid \mathcal{D})}{q(\theta)} q(\theta) d\theta, \text{ to avoid infinite importance weights} \\
&\approx \frac{1}{N} \sum_{n=1}^{N} \frac{p\left(\hat{\theta}_n \mid \mathcal{D}\right)}{q\left(\hat{\theta}_n\right)} f\left(\hat{\theta}_n\right) \quad \text{where} \quad \hat{\theta}_n \sim q(\theta) = \frac{1}{N} \sum_{n=1}^{N} w_n f(\hat{\theta}_n)
\end{aligned}$$

• Both unbiased and consistent (i.e. as $N \to \infty$ the estimate converges to the true expectation) subject to some mild assumptions (i.e. the proposal has heavier tails than the posterior).
• **Self–Normalized Importance Sampling** (SNIS): when posterior cannot be computed explicitly, we update the weights using the unnormalized density: $\tilde{w}_n = \frac{P(\theta,\mathcal{D})}{q(\theta)}$, then
$\mathbb{E}_{p(\theta|\mathcal{D})}[f(\theta)] \approx \sum_{n=1}^N \bar{w}_n f\left(\hat{\theta}_n\right)$, where $\bar{w}_n = \frac{\tilde{w}_n}{\sum_n \tilde{w}_n}$, because it is the ratio of $^{(ii)}/_{(i)}$:
(i) $\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^N w_n\right] = \frac{1}{N}\sum_{n=1}^N \mathbb{E}\left[w_n\right] = \mathbb{E}_{q(\hat{\theta}_1)}\left[\frac{p(\hat{\theta}_1,\mathcal{D})}{q(\hat{\theta}_1)}\right] = p(\mathcal{D})$ and
(ii) $\mathbb{E}_{q(\theta)}\left[\frac{p(\theta,\mathcal{D})}{q(\theta)} f(\theta)\right] = E_{q(\theta)}\left[\frac{p(\theta|\mathcal{D})}{q(\theta)} p(\mathcal{D}) f(\theta)\right] = p(\mathcal{D})\mathbb{E}_{p(\theta|\mathcal{D})}[f(\theta)]$

3.4 Gaussian Process (GP)

4 Types of Regression:
(i) **Frequentist Parametric approach**: model f as f_θ for some parameter vector θ. Fit θ by ML/ERM with squared loss, e.g. linear regression.
(ii) **Frequentist Nonparametric approach**: model f as the unknown parameter taking values in an infinite-dimensional space of functions (RKHS). Fit f by regularized ML/ERM with squared loss, e.g. kernel ridge regression.
(iii) **Bayesian Parametric approach**: model f as f_θ for some parameter vector θ. Put a prior on θ and compute a posterior $P(\theta|\mathcal{D})$, e.g. Bayesian linear regression.
(iv) **Bayesian Nonparametric approach**: treat f as the random variable taking values in an infinite-dimensional space of functions. Put a prior over functions $f \in \mathcal{F}$, and compute a posterior $P(f|\mathcal{D})$, e.g. Gaussian Process regression.

3.4.1 GP Regression

Gaussian Process (GP): $\forall (x_1, \cdots, x_n), \mathbf{f} = [f(x_1), \cdots, f(x_n)]^T \sim N(\mathbf{m}, \mathbf{K})$, where $\mathbf{m}_i = m(x_i) = \mathbb{E}[f(x_i)]$ and $\mathbf{K}_{ij} = k\left(x_i, x_j\right) = \mathbb{E}\left[\left(f(x_i) - m(x_i)\right)\left(f\left(x_j\right) - m\left(x_j\right)\right)\right]$
• The prior of mean $\mathbf{m(x)}$ is often set to be zero, and the covariance function $k(x, x')$ is positive definite hence a kernel.

GP Regression Model:

$$\begin{aligned} \mathbf{f} &\sim \mathcal{N}(0, \mathbf{K}) \\ \mathbf{y} \mid \mathbf{f} &\sim \mathcal{N}\left(\mathbf{f}, \sigma^2 I\right) \end{aligned}$$

$\implies \begin{bmatrix} \mathbf{f} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K} \\ \mathbf{K} & \mathbf{K} + \sigma^2 I \end{bmatrix}\right)$, because:
(i) $\mathbb{E}\left[\mathbf{f}\mathbf{y}^\top\right] = \mathbb{E}\left[\mathbf{f}(\mathbf{f} + \sigma\epsilon)^\top\right] = \underbrace{\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right]}_{=\mathbf{K}} + \underbrace{\sigma\mathbb{E}\left[\mathbf{f}\epsilon^\top\right]}_{=0} = \mathbf{K}$
(ii) $\mathbb{E}[\mathbf{y}\mathbf{y}^T] = \mathbb{E}\left[(\mathbf{f} + \sigma\epsilon)(\mathbf{f} + \sigma\epsilon)^\top\right] = \mathbb{E}[\mathbf{f}\mathbf{f}^T] + \sigma^2 \underbrace{\mathbb{E}[\epsilon\epsilon^T]}_{=I \because \epsilon \sim N(0,I)} + 2\sigma\mathbb{E}[\mathbf{f}\epsilon^T] = \mathbf{K} + \sigma^2 I$
$\implies \mathbf{f} \mid \mathbf{y} \sim \mathcal{N}\left(\mathbf{K}\left(\mathbf{K} + \sigma^2 I\right)^{-1}\mathbf{y}, \mathbf{K} - \mathbf{K}\left(\mathbf{K} + \sigma^2 I\right)^{-1}\mathbf{K}\right)$, by Gaussian conditioning.

Gaussian Conditioning: Let $\mathbf{z} \sim N(\mu, \Sigma)$ s.t. $\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}$, $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$, $\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$,

where $\Sigma_{12} = \Sigma_{21}^T$ by symmetry. Then:

$$p\left(\mathbf{z}_2 \mid \mathbf{z}_1\right) = \mathcal{N}\left(\mathbf{z}_2; \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}\left(\mathbf{z}_1 - \mu_1\right), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right)$$

Posterior Predictive Distribution:
Given the joint normal model:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}' \end{bmatrix} \mid \mathbf{x}, \mathbf{x}' \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{K_{xx}} & \mathbf{K_{xx'}} \\ \mathbf{K_{x'x}} & \mathbf{K_{x'x'}} \end{bmatrix}\right)$$

$$\mathbf{y} \mid \mathbf{f} \sim \mathcal{N}\left(\mathbf{f}, \sigma^2 I\right)$$

$$\implies \begin{bmatrix} \mathbf{f}' \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{K_{x'x'}} & \mathbf{K_{x'x}} \\ \mathbf{K_{xx'}} & \mathbf{K_{xx}} + \sigma^2 I \end{bmatrix}\right)$$

$\implies$ The predictive distribution: $\mathbf{f}' \mid \mathbf{y} \sim \mathcal{N}\left(\mathbf{K_{x'x}}\left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1}\mathbf{y}, \mathbf{K_{x'x'}} - \mathbf{K_{x'x}}\left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1}\mathbf{K_{xx'}}\right)$

Even if we do not have the joint normal observation model, we can still reason the predictive distribution by: $p(\mathbf{f}'|\mathbf{y}) = \int p(\mathbf{f}'|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f}$, with the property: $\int N(a; Bc, D)N(c; e, F)dc = N(a; Be, D + BFB^T)$. proof?

Gaussian Process Posterior: A GP prior $f \sim GP(0, k_{\text{prior}})$ conjugated to the Gaussian likelihood $\mathbf{y} \mid \mathbf{f} \sim \mathcal{N}\left(\mathbf{f}, \sigma^2 I\right)$ leads to the GP posterior:

$$f \mid \mathbf{y} \sim \text{GP}\left(m_{\text{post}}\ , k_{\text{post}}\ \right)$$

where:
(i) $m_{\text{post}}\ (x) = \mathbf{K_x}(x)^T\left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1}\mathbf{y}$
(ii) $k_{\text{post}}\ (x, x') = k_{\text{prior}}\ (x, x') - \mathbf{K_x}(x)^T\left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1}\mathbf{K_x}\left(x'\right)$
(iii) $\mathbf{K_x}(x) = \begin{bmatrix} k_{\text{prior}}\ (x, x_1) & \ldots & k_{\text{prior}}\ (x, x_N) \end{bmatrix}^T$
Kernel Ridge Regression (KRR) vs Gaussian Process Regression (GPR): KRR estimate of the function coincides with the GPR posterior mean, if the regularization parameter λ in KRR equals the noise variance σ^2 in GPR.
By Representer Thm, the solution of KRR problem:

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n}\left(y_i - f\left(x_i\right)\right)^2 + \sigma^2\|f\|_{\mathcal{H}_k}^2 \implies f(x) = \sum_{i=1}^{n}\alpha_i k(\cdot, x_i)$$

where $\alpha = \left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1}\mathbf{y}$. proof? why??
Hence, for a new set of input $\{x_j\}_{j=1}^m$, we have the KRR prediction equals the GPR posterior predictive mean: $f\left(x_j'\right) = \sum_{i=1}^n \alpha_i k\left(x_j', x_i\right) = \underbrace{\left[k\left(x_j', x_1\right), \ldots, k\left(x_j', x_n\right)\right]}_{[\mathbf{K_{x'x}}]_{j,:}}\left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1}\mathbf{y}$

Hyperparameter Selection: By maximizing marginal likelihood.

The marginal likelihood of $\theta = (\nu, \sigma^2)$: $p(\mathbf{y} \mid \theta) = \int p(\mathbf{y} \mid \mathbf{f}, \theta)p(\mathbf{f} \mid \theta)d\mathbf{f} = \mathcal{N}\left(\mathbf{y}; 0, \underbrace{\mathbf{K}_\nu + \sigma^2 I}_{\mathbf{K}_{\theta+}}\right)$.

$\implies \log p(\mathbf{y} \mid \theta) = -\frac{1}{2}\log|\mathbf{K}_{\theta+}| - \frac{1}{2}\mathbf{y}^\top\mathbf{K}_{\theta+}^{-1}\mathbf{y} - \frac{n}{2}\log(2\pi)$, a nonconvex function which may have multiple maxima.
Use numerical optimisation methods (e.g. gradient ascent), with $\frac{\partial}{\partial\theta_i}\log p(\mathbf{y} \mid \theta) = -\frac{1}{2}\text{Tr}\left(\mathbf{K}_{\theta+}^{-1}\frac{\partial\mathbf{K}_{\theta+}}{\partial\theta_i}\right) + \frac{1}{2}\mathbf{y}^\top\mathbf{K}_{\theta+}^{-1}\frac{\partial\mathbf{K}_{\theta+}}{\partial\theta_i}\mathbf{K}_{\theta+}^{-1}\mathbf{y}$.

3.4.2 GP Classification

Transform the GP output wth an activation function so that the output $\in [0,1]$, e.g. logistic sigmoid $p(y_i = +1|f(x_i)) = \frac{1}{1+e^{-f(x_i)}}$. (non-Gaussian form likelihood)
The posterior:

$$\begin{aligned}\log p(\mathbf{f} \mid \mathbf{y}) &= \text{ const } + \log p(\mathbf{f}) + \log p(\mathbf{y} \mid \mathbf{f}) \\ &= \text{ const } - \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + \sum_{i=1}^{n} \log s\left(y_i f\left(x_i\right)\right)\end{aligned}$$

Can be approximated via Laplace Approximation, with the gradients and Hessian:

$$\begin{aligned}&\frac{\partial \log p(\mathbf{f} \mid \mathbf{y})}{\partial \mathbf{f}} = -\mathbf{K}^{-1}\mathbf{f} + \mathbf{g_f}, \quad [\mathbf{g_f}]_i = \frac{\partial \log p(\mathbf{y} \mid \mathbf{f})}{\partial f_i} = sig(-y_i f(x_i))y_i \\ &\frac{\partial^2 \log p(\mathbf{f} \mid \mathbf{y})}{\partial \mathbf{f}\partial \mathbf{f}^\top} = -\mathbf{K}^{-1} - \mathbf{D}_f, \quad [\mathbf{D}_f]_{ii} = \frac{\partial^2 \log p(\mathbf{y} \mid \mathbf{f})}{\partial f_i^2} = sig(f(x_i))sig(-f(x_i)), \quad [\mathbf{D}_f]_{ij} = 0 \\ &\implies \tilde{p}(\mathbf{f} \mid \mathbf{y}) = \mathcal{N}\left(\mathbf{f} \mid \hat{\mathbf{f}}^{\text{MAP}}, \left(\mathbf{K}^{-1} + \mathbf{D}_{\hat{\mathbf{f}}^{\text{MAP}}}\right)^{-1}\right) = \mathcal{N}\left(\mathbf{f} \mid \hat{\mathbf{f}}^{\text{MAP}}, \mathbf{K} - \mathbf{K}\left(\mathbf{K} + \mathbf{D}^{-1}_{\hat{\mathbf{f}}^{\text{MAP}}}\right)^{-1}\mathbf{K}\right)\end{aligned}$$

by Woodbury identity: $(\mathbf{K} + \mathbf{D})^{-1} = \mathbf{K} - \mathbf{K}(\mathbf{K} + \mathbf{D})^{-1}\mathbf{K}$, for invertible matrices $\mathbf{K}, \mathbf{D}$.

And the predictive distribution:

$$\tilde{p}\left(\mathbf{f}' \mid \mathbf{y}\right) = \int p\left(\mathbf{f}' \mid \mathbf{f}\right)\tilde{p}(\mathbf{f} \mid \mathbf{y})d\mathbf{f} = \mathcal{N}\left(\mathbf{f}' \mid \mathbf{K}_{\mathbf{x}'\mathbf{x}}\mathbf{K}_{\mathbf{xx}}^{-1}\hat{\mathbf{f}}^{\text{MAP}}, \mathbf{K}_{\mathbf{x}'\mathbf{x}'} - \mathbf{K}_{\mathbf{x}'\mathbf{x}}\left(\mathbf{K}_{\mathbf{xx}} + \mathbf{D}^{-1}_{\hat{\mathbf{f}}^{\text{MAP}}}\right)^{-1}\mathbf{K}_{\mathbf{xx}'}\right)$$

where $p\left(\mathbf{f}' \mid \mathbf{f}\right) = \mathcal{N}\left(\mathbf{f}' \mid \mathbf{K}_{\mathbf{x}'\mathbf{x}}\mathbf{K}_{\mathbf{xx}}^{-1}\mathbf{f}, \mathbf{K}_{\mathbf{x}'\mathbf{x}'} - \mathbf{K}_{\mathbf{x}'\mathbf{x}}\mathbf{K}_{\mathbf{xx}}^{-1}\mathbf{K}_{\mathbf{xx}'}\right)$

Computing $\hat{\mathbf{f}}^{MAP}$: via Newton-Raphson, it can be numerically unstable when $\mathbf{K}$ is small, woodbury identity saves us for avoiding to inverse $\mathbf{K}$.

$$\begin{aligned}\mathbf{f}^{\text{new}} &= \mathbf{f} - \left(\frac{\partial^2 \log p(\mathbf{f} \mid \mathbf{y})}{\partial \mathbf{f}\partial \mathbf{f}^\top}\right)^{-1}\frac{\partial \log p(\mathbf{f} \mid \mathbf{y})}{\partial \mathbf{f}} \\ &= \mathbf{f} + \left(\mathbf{K}^{-1} + \mathbf{D}_f\right)^{-1}\left(\mathbf{g_f} - \mathbf{K}^{-1}\mathbf{f}\right) \\ &= \left(\mathbf{K}^{-1} + \mathbf{D}_f\right)^{-1}\left(\mathbf{K}^{-1} + \mathbf{D}_f\right)\mathbf{f} + \left(\mathbf{K}^{-1} + \mathbf{D}_f\right)^{-1}\left(\mathbf{g_f} - \mathbf{K}^{-1}\mathbf{f}\right) \\ &= \left(\mathbf{K}^{-1} + \mathbf{D}_f\right)^{-1}\left(\mathbf{D}_f\mathbf{f} + \mathbf{g_f}\right) \\ &= \left[\mathbf{K} - \mathbf{K}\left(\mathbf{K} + \mathbf{D}_{\mathbf{f}}^{-1}\right)^{-1}\mathbf{K}\right]\left(\mathbf{D}_f\mathbf{f} + \mathbf{g_f}\right), \text{ by Woodbury}\end{aligned}$$

3.4.3 Large-Scale Kernel Approximations

GP and Kernel methods are computationally expensive ($\mathcal{O}(n^2) \sim \mathcal{O}(n^3)$), due to compute/store/invert $\mathbf{K}$. Hence we look forward to the *reduced-rank approximation* of $\mathbf{K_{xx}}$.

Low Rank Matrix Approximations (Nyströrm Approximation):

$$\tilde{\mathbf{K}}_{\mathbf{xx}} = \mathbf{K}_{\mathbf{xz}}\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{K}_{\mathbf{zx}} = QQ^T, \quad Q = \mathbf{K}_{\mathbf{xz}}\mathbf{K}_{\mathbf{zz}}^{-1/2}$$

where the **inducing points** $\{z_j\}_{j=1}^m$ is a subset of the training set (a small number of input in $\mathcal{X}$).
$\implies (\mathbf{K_{xx}} + \sigma^2 I)^{-1}$ can be approximated by $\left(QQ^\top + \sigma^2 I\right)^{-1} = \sigma^{-2}I - \sigma^{-2}Q\left(\sigma^2 I + Q^\top Q\right)^{-1}Q^\top$

Random Fourier Features:
If the kernel only depends on the difference between datapoints, then the Fouier representaton is:

$$k\left(x, x'\right) = \kappa(x - x') = 2\kappa(0)\mathbb{E}\left[\cos\left(\omega^\top x + b\right)\cos\left(\omega^\top x' + b\right)\right]$$
$$\approx k_m\left(x, x'\right) := \frac{2\kappa(0)}{m}\sum_{j=1}^{m}\cos\left(\hat{\omega}_j^\top x + \hat{b}_j\right)\cos\left(\hat{\omega}_j^\top x' + \hat{b}_j\right)$$

where $b \sim Unif(0, 2\pi)$ and $\omega \in \mathbb{R}^p$ has density given by the normalized Fourier transform of κ:

$$\omega \sim p(\omega) \propto \int_{\delta\in\mathbb{R}^p}\kappa(\delta)\exp\left(-i\omega^T\delta\right)d\delta = \int_{\delta\in\mathbb{R}^p}\kappa(\delta)\left[\cos(\omega^T\delta) - i\sin(\omega^T\delta)\right]d\delta, \text{ by Euler's}$$
$$= \int_{\delta\in\mathbb{R}^p}\kappa(\delta)\cos\left(\omega^T\delta\right)d\delta, \quad \because \sin(\cdot) \text{ asymmetric around } 0$$

• Note that the approximator can be viewed as a inner product between feature maps $\varphi_m : \mathbb{R}^p \to \mathbb{R}^m$, where $\varphi_m(x) = \sqrt{\frac{2\kappa(0)}{m}}\left[\cos\left(\hat{\omega}_1^\top x + \hat{b}_1\right), \cos\left(\hat{\omega}_2^\top x + \hat{b}_2\right), \ldots, \cos\left(\hat{\omega}_m^\top x + \hat{b}_m\right)\right]^\top$

3.5 Bayesian Optimization (BO)

BO deals with "Black-Box" models, namely, point-wise output evaluations for given input.
2 Key problems:
(i) Where to *evaluate* the function;
• **Exploration–Exploitation trade–off**: evaluate points where our uncertainty is high to reduce the uncertainty in those regions, versus, evaluate points where the expected function value is high so that we get a good characterization of the function in promising regions.
- **Acquisition function** $\zeta : \mathcal{X} \to \mathbb{R}$ came into place here.
(ii) Where we *predict* the optimum to be given our evaluations. • Need to marginalize out the noise introduced by GP:
- return of a point that is not actually in our set of evaluations at all by finding and returning $\arg\max_{x\in\mathcal{X}}\mu(x)$; or
- choose the evaluated point with the best lower bound on the function by considering $\mu(x) - \beta\sigma(x)$ for some $\beta \geq 0$.

Optimization Through a Surrogate Model: GP Surrogate
Consider GP $\sim N(\mu(x), \sigma(x) := \sqrt{k_{post}(x, x)})$. read section 7.2 of the Advanced Topics in ML notes for details.

3.5.1 Acquisition function

Probability of Improvement (PI): Probability that the function value at a point is higher than the current estimated optimum, with marginal distributions for function evaluations being Gaussian.

$$\zeta_{\text{PI}}(x) := P\left(f(x) \geq \mu^+ + \xi \mid \mathcal{D}\right)$$
$$= \int_{\mu^+ + \xi}^{\infty}\mathcal{N}\left(f(x); \mu(x), \sigma^2(x)\right)df(x)$$
$$= \Phi(\gamma(x))$$

where $\mu^+ = \max_{i\in\{1,\dots n\}} \mu(x_i)$ (the point with the highest expected value), $\gamma(x) = \frac{\mu(x)-\mu^+-\xi}{\sigma(x)}$, $\xi \geq 0$ is the threshold of improvement, and $\Phi(\cdot)$ the standard Normal CDF.
• $\xi \to 0 \implies$ pure exploitation, choosing to evaluate the points with small $\sigma(x)$;
• $\xi \to \infty \implies$ pure exploration, choosing to evaluate the points with large $\sigma(x)$.
• Very sensitive to ξ, and failed to encapsulate the magnitude of any potential gains, hence PI is rarely used in practice.

Expected Improvement

$$\begin{aligned}
\zeta_{\text{EI}}(x) &:= \int_{\mu^+ + \xi}^{\infty} \left(f(x) - \mu^+ - \xi\right) \mathcal{N}\left(f(x); \mu(x), \sigma^2(x)\right) df(x) \\
&= \int_0^{\infty} s\mathcal{N}\left(s + \mu^+ + \xi; \mu(x), \sigma^2(x)\right) ds \\
&= \int_0^{\infty} s\mathcal{N}\left(s; \mu(x) - \mu^+ - \xi, \sigma^2(x)\right) ds \\
&= \left(\mu(x) - \mu^+ - \xi\right) \Phi(\gamma(x)) + \sigma(x)\mathcal{N}(\gamma(x); 0, 1), \\
\text{by } \mathbb{E}[\max(0, S)] &= \int_0^{\infty} s\mathcal{N}\left(s; m, \tau^2\right) ds = m\Phi\left(\frac{m}{\tau}\right) + \tau\mathcal{N}\left(\frac{m}{\tau}; 0, 1\right), \quad S \sim N(m, \tau^2) \\
&= \sigma(x)(\gamma(x)\Phi(\gamma(x)) + \mathcal{N}(\gamma(x); 0, 1))
\end{aligned}$$

• ξ place a similar role in EI as in PI, despite EI is less sensitive to ξ.

Upper Confidence Bounding (UCB): $UCB(x) = \mu(x) + \beta\sigma(x)$, with the **optimism boost** controls the exploration-exploitation trade-off.
• UCB represents the point at which the probability the function is less than this value is $\Phi(\beta)$.
• Large β encourages exploration to the region with higher uncertainty.

Information-Based Policies (IBP): Instead of maximizing the acquisition function, IBP instead learns the distribution of the location of the maximum $P(x^*|\mathcal{D})$ given the previous evaluation pairs, and sample the next-point-to-evaluate from $P(x^*|\mathcal{D})$. see Advanced Topics in ML notes for details.

3.6 Deep Learning

3.6.1 DL Basics

Neural Network:

$$\begin{aligned}
\text{Input layer: } & h^0 = x \\
\text{Hidden layer: } & h^\ell = f^{(\ell)}\left(h^{\ell-1}\right) \quad \text{for } \ell = 1, \dots, m-1 \\
\text{Output layer: } & h^m = x \\
& f(x) = h^m = f^{(m)} \circ f^{(m-1)} \circ \cdots f^{(2)} \circ f^{(1)}(x)
\end{aligned}$$

where $f = (f^{(l)})_{l=1}^m$ is differentiable and parametrized by $\theta = (\theta^{(l)})_{l=1}^m$ of lengths $p = \sum_{l=1}^m p_l$.

The ERM problem: $\min_\theta \hat{R}(\theta) = \lambda r(\theta) + \frac{1}{n}\sum_{i=1}^n L\left(y_i, f_\theta\left(x_i\right)\right)$ (with $r(\cdot)$ being the regularizer)
• Apply gradient descent: $\theta_t = \theta_{t-1} - \alpha\nabla_\theta \hat{R}\left(\theta_{t-1}\right) = \theta_{t-1} - \alpha\lambda\nabla_\theta r(\theta) + \frac{1}{n}\sum_{i=1}^n \nabla_\theta L_i$, where

$L_i = L\left(y_i, f_\theta\left(x_i\right)\right)$.

Gradient of the loss can be computed by recursively applying the chain rule:

$$\frac{\partial L_i}{\partial \theta^{(\ell)}} = \frac{\partial L_i}{\partial h_i^m} \frac{\partial h_i^m}{\partial h_i^{m-1}} \cdots \frac{\partial h_i^{\ell+1}}{\partial h_i^\ell} \frac{\partial h_i^\ell}{\partial \theta^{(\ell)}}$$

where:
- $\frac{\partial L_i}{\partial h_i^m}$ is a $d_m \times 1$ vector representing the derivative of loss wrt each unit of the output layer;
- $\frac{\partial h_i^k}{\partial h_i^{k-1}}$ is a $d_k \times d_{k-1}$ matrix representing the Jacobian of the $k-$th hidden layer wrt to the precedent hidden layer;
- $\frac{\partial h_i^\ell}{\partial \theta^{(\ell)}}$ is a $d_l \times p_l$ matrix representing the Jacobian of the $l-$th hidden layer wrt its parameters.

Backpropagation: Compute gradients of the loss with respect to the variables and parameters in a *backward* fashion, i.e. the first line, because it is computationally cheaper.

$$\begin{aligned}
\frac{\partial L_i}{\partial \theta^{(\ell)}} &= \underbrace{\left(\left(\left(\frac{\partial L_i}{\partial h_i^m} \frac{\partial h_i^m}{\partial h_i^{m-1}}\right) \cdots \frac{\partial h_i^{\ell+1}}{\partial h_i^\ell}\right) \frac{\partial h_i^\ell}{\partial \theta^{(\ell)}}\right)}_{O(d_m d_{m-1} + d_{m-1} d_{m-2} + \cdots + d_{\ell+1} d_\ell + d_\ell p_\ell) = O\left(d_\ell p_\ell + \sum_{k=\ell+1}^m d_k d_{k-1}\right)} \\
&= \underbrace{\left(\frac{\partial L_i}{\partial h_i^m}\left(\frac{\partial h_i^m}{\partial h_i^{m-1}} \cdots \left(\frac{\partial h_i^{\ell+1}}{\partial h_i^\ell} \frac{\partial h_i^\ell}{\partial \theta^{(\ell)}}\right)\right)\right)}_{O(d_{\ell+1} d_\ell p_\ell + d_{\ell+2} d_{\ell+1} p_\ell + \cdots + d_m d_{m-1} p_\ell + d_m p_\ell) = O\left(d_m p_\ell + \sum_{k=\ell+1}^m d_k d_{k-1} p_\ell\right)}
\end{aligned}$$

• The gradient wrt all parameters θ can be computed in a single backward propagation phase, by computing $\frac{\partial L_i}{\partial h_i^\ell} = \frac{\partial L_i}{\partial h_i^{\ell+1}} \frac{\partial h_i^{\ell+1}}{\partial h_i^\ell}$ and $\frac{\partial L_i}{\partial \theta^{(\ell)}}$ iteratively, for $l = m-1, ..., 1$.

Computation Graphs: For inputs u_i and outputs v_i of an operation $(v_1, \ldots, v_n) = \text{ op } (u_1, \ldots, u_m)$,
(i) Backward mode differentiation (**vector-Jacobian product**): $\text{vjp}\left(\left(u_i\right)_{i=1}^m, \left(\Delta_j\right)_{j=1}^n\right) = \left(\sum_{j=1}^n \Delta_j \cdot \frac{\partial v_j}{\partial u_i}\right)_{i=1}^m$
(ii) Forward mode differentiation (**Jacobian-vector product**): $\text{jvp}\left(\left(u_i\right)_{i=1}^m, \left(\delta_i\right)_{i=1}^m\right) = \left(\sum_{i=1}^n \frac{\partial v_j}{\partial u_i} \cdot \delta_i\right)_{j=1}^n$
where Δ_i (δ_i) is a tensor of the same shape as v_i (u_i), interpreted as the gradient of some objective function (of u_i) wrt v_i (some scalar).
• The overall computational cost for forward mode differentiation is linear in the dimensionality of the inputs of the graph, while backward mode differentiation has computational cost linear in the dimensionality of the outputs. In machine learning the final output is almost always a single dimension (the objective) so backward mode differentiation is much more predominant.

3.6.2 Modules

op: A *node* in a computation graph. Given an input, its output will be a block of (hidden) unit values. We can think of ops as evaluations of functions; they have no parameters of their own.
module: A *function* that can applied multiple times within the graph. It may be parameterised, in which case the parameters are shared across uses. Applying a module creates an op.
factory: A procedure that *generates* modules. This allows us to produce multiple modules that have separate sets of parameters.
- Factory(n, σ) generates functions $f : \mathbb{R}^n \to \mathbb{R}^n$ of the form $f(x) = \sigma(Wx)$, with $W \in \mathbb{R}^{n \times n}$ and

σ is the element-wise non-linerity.

Linear Model:

$$\begin{aligned} \text{module} &\sim \text{Linear}\,(m,n) \\ \text{module}\,(x) &= Wx+b \end{aligned}$$

Non-linearity functions: all element-wise, except for softmax

$$\begin{aligned} \text{sigmoid}(x) &= 1/(1+\exp(-x)) \\ \tanh(x) &= (\exp(x)-\exp(-x))/(\exp(x)+\exp(-x)) \\ \text{ReLU}(x) &= \max(0,x) \\ \text{softplus}(x) &= \log(1+\exp(x)) \\ \text{swish}(x) &= x\,\text{sigmoid}(x) \\ \text{ELU}_\alpha(x) &= \begin{cases} x & \text{if } x>0 \\ \alpha(\exp(x)-1) & \text{if } x\le 0 \end{cases} \\ \text{softmax}\left([x_1,\ldots,x_d]\right) &= \left[\frac{\exp\left(x_1\right)}{\sum_{i=1}^d \exp\left(x_i\right)},\cdots,\frac{\exp\left(x_d\right)}{\sum_{i=1}^d \exp\left(x_i\right)}\right] \end{aligned}$$

2D Convolutional module:

$$\begin{aligned} \text{module} &\sim \text{Conv 2D}\left(c_{\text{in}},c_{\text{out}},d_1,d_2\right) \\ x' &= \text{module}(x) \\ x'_{i'j'k'} &= \sum_{i=1}^{d_1}\sum_{j=1}^{d_2}\sum_{k=1}^{c_{in}} w_{ijkk'}x_{i'+i-1,j'+j-1,k} \quad \forall i',j',k' \end{aligned}$$

where $\{w_{ijkk'}\}_{i=1:d_1,j=1:d_2}$ is the **filter**, a 2-dim array that is unique for each input-output channel pair (kk'), akin to the weights of a linear module.
• Variations: with-bias, padding, striding.

Max-pooling:

$$\begin{aligned} x' &= \text{MaxPool}_{d_1,d_2}(x) \\ x'_{i'j'k'} &= \max_{i=1}^{d_1}\max_{j=1}^{d_2} x_{i+d_1(i'-1),j+d_2(j'-1),k'} \end{aligned}$$

• Also, **Average-pooling**.

Multi-Layer Perceptrons (MLP):

$$\begin{aligned} f &\sim \text{MLP}\left(d_m,d_{m-1},\ldots,d_0,\sigma\right) = \begin{cases} f^{(\ell)} \sim \text{Linear}\left(d_\ell,d_{\ell-1}\right) \text{ for } \ell=1,\ldots,m \\ f = f^{(m)}\circ\sigma\circ f^{(m-1)}\circ\cdots\circ\sigma\circ f^{(1)} \end{cases} \\ f(x) &= W_m\sigma\left(W_{m-1}\sigma\left(\cdots W_2\sigma\left(W_1x+b_1\right)+b_2\cdots\right)+b_{m-1}\right)+b_m \end{aligned}$$

• One can also have various activation functions $(\sigma_l)_{l=1}^m$.

Recurrent Neural Network (RNN): Given sequence of input $x_1, x_2, ...$ and sequence of outputs $y_1, y_2, ...$, and the prediction of y_t depends on $x_{1:t}$.

$$\begin{aligned}
\text{encode} &\sim \text{EncoderFactory} \\
\text{decode} &\sim \text{DecoderFactory} \\
h_t &= \text{encode}\,(h_{t-1}, x_t) \\
\hat{y}_t &= \text{decode}\,(h_t)
\end{aligned}$$

Long short-term memory (LSTM): specifically to address gradient explosion/vanishing problems in standard RNNs.

$$\begin{aligned}
\text{forget_gate}_t &= \text{sigmoid}\,(W_f\,[h_{t-1}, x_t] + b_f) \\
\text{insert_gate}_t &= \text{sigmoid}\,(W_i\,[h_{t-1}, x_t] + b_i) \\
C'_t &= \tanh\,(W_C\,[h_{t-1}, x_t] + b_C) \\
C_t &= \text{forget_gate}_t * C_{t-1} + \text{insert_gate}_t * C'_t \\
\text{output_gate}_t &= \text{sigmoid}\,(W_o\,[h_{t-1}, x_t] + b_o) \\
h_t &= \text{output_gate}_t * \tanh\,(C_t)
\end{aligned}$$

where the **cell state** C_t represent the long-term memory and the hidden state h_t represent the short-term memory.
• Recall that the gradient of MLP is a product of a sequence of matrices, which can go very large (**exploding gradients**) or very small (**vanishing gradients**), if the matrix has large eigenvalues (> 1) or small eigenvalues (close to 0).

3.6.3 Initialisation and Regularization

Xavier initialisation: Consider an MLP $\tilde{h}_j^\ell = \sum_{k=1}^{d_{\ell-1}} W_{jk}^\ell h_k^{\ell-1} + b_j^\ell$, control the scale of $\tilde{h}_j^\ell$ to be $\mathcal{O}(1)$ by setting $\sigma_{\ell w}^2 = \mathcal{O}(1/d_{\ell-1}) = 1/d_{\ell-1}$.
• Because $Var(\tilde{h}_j^\ell) = \mathcal{O}(d_{\ell-1}\sigma_{\ell w}^2 + \sigma_{\ell b}^2)$, with $\sigma_{\ell w}^2$ and $\sigma_{\ell b}^2$ being the variance of the weight and bias respectively.

Regularizations:
(i) L_2- norm can be thought of as a **weight decay**, as the parameter is decaied by $(1 - \eta_t\lambda)$ at each iteration.
(ii) **Stochasticity** in SGD.
(iii) **Dropout**: During training, for each data item, and for each unit in the network, randomly remove it from the network (with probability p, typically p = 0.5) by multiplying its activation by 0.
(iv) **Early stopping**: Monitor the validation loss during training, and stop training when it starts going up.

3.7 Latent Variable Models (LVM)

Goal: Model the data generating process using latent variables, with the latent representation as a summary of the data.

3.7.1 LVM Basics, Mixture Modelling, and KL Divergence

Latent Variable Models (LVM): a probabilistic generative model where each datapoint x_i has a corresponding latent variable z_i.

$$\begin{aligned} p_\theta(\mathbf{X}, \mathbf{Z}) &= p_\theta(\mathbf{Z}) \prod_{i=1}^{n} p_\theta\left(x_i \mid z_i\right) \\ &= \prod_{i=1}^{n} p_\theta(z_i) p_\theta\left(x_i \mid z_i, \theta\right), \text{ under cond. indep. assumption of } z_i \text{ given } \theta \\ p(\theta, \mathbf{X}, \mathbf{Z}) &= p(\theta) p(\mathbf{Z}|\theta) \prod_{i=1}^{n} p_\theta\left(x_i \mid z_i, \theta\right), \text{ if Bayesian} \end{aligned}$$

where θ is the global variable.

Mixture model: Assume that our dataset $\mathbf{X}$ was created by sampling iid from K distinct populations (called **mixture components**), where each population k follows F_k with density $f_k(x; \theta_k)$.

Algorithm 14: Mixture Modelling

Initialize K population distributions $f_k(x; \theta_k)$ for the K components.
for $i = 1, ..., n$ **do**
  Independently sample **assignment variable**: $Z_i \sim Discrete(\pi_1, \cdots, \pi_k)$ with **mixture weights** $P(Z_i = k) = \pi_k$ and $\sum_{k=1}^{K} \pi_k = 1$.
  Independently sample $X_i = (X_i^{(1)}, \cdots, X_i^{(p)})^T \mid Z_i = k \sim f_k(x; \theta_k)$
end
Inference: computation or approximation of the posterior distribution,
$p_\theta(\mathbf{Z} \mid \mathbf{X}) = \frac{p_\theta(\mathbf{X}, \mathbf{Z})}{p_\theta(\mathbf{X})} = \prod_{i=1}^{n} p_\theta\left(z_i \mid x_i\right)$, with,
• joint density is: $p_\theta(\mathbf{X}, \mathbf{Z}) = \prod_{i=1}^{n} \pi_{z_i} f_k\left(x_i; \theta_{z_i}\right)$
• marginal likelihood:
$p_\theta(\mathbf{X}) = \sum_{z_1=1}^{K} \cdots \sum_{z_n=1}^{K} \prod_{i=1}^{n} \pi_{z_i} f_k\left(x_i; \theta_{z_i}\right) = \prod_{i=1}^{n} \left(\sum_{k=1}^{K} \pi_k f_k\left(x_i; \theta_k\right)\right)$
Learning: estimation of the model parameters θ via maximising the marginal likelihood.

.

Kullback- Leibler (KL) divergence (relative entropy):

$$\mathbb{D}_{\mathrm{KL}}(P \| Q) = \mathbb{E}_p\left(\log \frac{p(X)}{q(X)}\right) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)}$$

• KL Divergence is positive because: $\mathbb{D}_{\mathrm{KL}}(P \| Q) = \mathbb{E}_p\left[-\log \frac{q(X)}{p(X)}\right] \geq -\log \mathbb{E}_p \frac{q(X)}{p(X)} = 0$, by convexity of $-\log(x)$ and Jensen's Inequality.

Jensen's Inequality: $\mathbb{E}[f(X)] \geq f(\mathbb{E}X)$ for convex function $f(X)$. (If $f(X)$ is strictly convex, then equality holds iff X is almost surely a constant).

3.7.2 Expectation Maximization (EM) Algorithm

Goal: maximise the marginal log-likelihood $\ell(\theta) = \log p_\theta(\mathbf{X}) = \log \int p(\mathbf{X}, \mathbf{Z}) d\mathbf{Z}$.

Variational free energy: $\mathcal{L}(\theta, q) = \mathbb{E}_{\mathbf{Z}\sim q}[\log p_\theta(\mathbf{X}, \mathbf{Z}) - \log q(\mathbf{Z})]$, where $q(\mathbf{Z})$ is the variational distribution over the latent variable $\mathbf{Z}$.
• This is same as **ELBO**, i.e. the lower bound of the log marginal likelihood.
Proof: Consider the KL divergence between the variational and the true conditional (knowing that it is non-negative):

$$0 \leq \mathbb{D}_{\mathrm{KL}}[q(\mathbf{Z})\|p_\theta(\mathbf{Z} \mid \mathbf{X})] = \mathbb{E}_{\mathbf{Z}\sim q}\left[\log \frac{q(\mathbf{Z})}{p_\theta(\mathbf{Z} \mid \mathbf{X})}\right] = \log p_\theta(\mathbf{X}) + \mathbb{E}_{\mathbf{Z}\sim q}\left[\log \frac{q(\mathbf{Z})}{p_\theta(\mathbf{X}, \mathbf{Z})}\right]$$

$$\implies \ell(\theta) = \log p_\theta(\mathbf{X}) \geq \mathbb{E}_{\mathbf{Z}\sim q}\left[\log \frac{p_\theta(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})}\right] = \underbrace{\mathbb{E}_{\mathbf{Z}\sim q}[\log p_\theta(\mathbf{X}, \mathbf{Z})]}_{\text{energy}} \overbrace{-\mathbb{E}_{\mathbf{Z}\sim q}[\log q(\mathbf{Z})]}^{\text{Shannon entropy}}$$

Since the Shannon entropy is independent of θ, one can think of it as the complexity penalty for q.

• Let $\mathcal{L}$ be the variational free energy in a latent variable model $p_\theta(\mathbf{X}, \mathbf{Z})$, then:
(a) Lower bound: $\mathcal{L}(\theta, q) \leq \ell(\theta), \forall(q, \theta)$;
(b) $\forall\theta, \mathcal{L}(\theta, q) = \ell(\theta) \iff q(\mathbf{Z}) = p_\theta(\mathbf{Z}|\mathbf{X})$.

Algorithm 15: EM Algorithm

Initialize $\theta^{(0)}$ and $t = 1$.
while *not converge* **do**
  E-step: Set $q^{(t)}(\mathbf{Z}) = p_{\theta^{(t-1)}}(\mathbf{Z} \mid \mathbf{X})$
  M-step: $\theta^{(t)} = \arg\max_\theta \mathbb{E}_{\mathbf{Z}\sim q^{(t)}}[\log p_\theta(\mathbf{X}, \mathbf{Z})]$
  $t = t + 1$
end

• Given that the variational free energy is a lower bound of the marginal likelihood $p(\mathbf{X})$, we can maximize the marginal likelihood by maximizing its lower bound, according to coordinate ascent.

EM for Mixture Modelling: Consider the mixture model above, $p_\theta(\mathbf{X}, \mathbf{Z}) = \prod_{i=1}^n \pi_{z_i} f_k(x_i; \lambda_{z_i})$. The variational free energy is:

$$\begin{aligned}
\mathcal{L}(\theta, q) &= \mathbb{E}_q[\log p_\theta(\mathbf{X}, \mathbf{Z}) - \log q(\mathbf{Z})] \\
&= \mathbb{E}_q\left[\left(\sum_{i=1}^n \sum_{k=1}^K \mathbb{I}(z_i = k)(\log \pi_k + \log f_k(x_i; \lambda_k))\right) - \log q(\mathbf{Z})\right] \\
&= \sum_{\mathbf{Z}} q(\mathbf{Z})\left[\left(\sum_{i=1}^n \sum_{k=1}^K \mathbb{I}(z_i = k)(\log \pi_k + \log f_k(x_i; \lambda_k))\right) - \log q(\mathbf{Z})\right] \\
&= \sum_{i=1}^n \sum_{k=1}^K \underbrace{q(z_i = k)}_{Q_{ik},\ \textbf{responsibility}} (\log \pi_k + \log f_k(x_i; \lambda_k)) + H(q)
\end{aligned}$$

Algorithm 16: EM for (Normal) Mixture Models with shared variance σ^2

Initialize K cluster means $\mu_1^{(0)}, \cdots, \mu_K^{(0)}$, mixing weights $\pi_1^{(0)}, \cdots, \pi_K^{(0)}$, and $t = 1$.

while *not converge* **do**

E-step: For fixed $\theta^{(t-1)} = (\lambda_{1:K}^{(t-1)}, \pi_{1:K}^{(t-1)})$

$$p_{\theta^{(t-1)}}(\mathbf{Z} \mid \mathbf{X}) = \frac{p_{\theta^{(t-1)}}(\mathbf{X}, \mathbf{Z})}{p_{\theta^{(t-1)}}(\mathbf{X})} = \frac{\prod_{i=1}^n \pi_{z_i}^{(t-1)} f_{z_i}\left(x_i; \lambda_{z_i}^{(t-1)}\right)}{\sum_{\mathbf{Z}'} \prod_{i=1}^n \pi_{Z_i'}^{(t-1)} f_{Z_i}\left(x_i; \lambda_{Z_i'}^{(t-1)}\right)}$$

$$= \prod_{i=1}^n \frac{\pi_{z_i}^{(t-1)} f_{z_i}\left(x_i; \lambda_{z_i}^{(t-1)}\right)}{\sum_k \pi_k^{(t-1)} f_k\left(x_i; \lambda_k^{(t-1)}\right)} = \prod_{i=1}^n p_{\theta^{(t-1)}}(z_i \mid x_i) = \prod_{i=1}^n \underbrace{Q_{i,z_i}^{(t)}}_{\textbf{responsible}}$$

where in mixture normal, $Q_{ik}^{(t)} = \frac{\pi_k \exp\left(-\frac{1}{2\sigma^2}\|x_i - \mu_k\|_2^2\right)}{\sum_{j=1}^K \pi_j \exp\left(-\frac{1}{2\sigma^2}\|x_i - \mu_j\|_2^2\right)}$

M-step: $\max_\theta \mathcal{L}(\theta, q)$ subject to $\sum_{k=1}^K \pi_k^{(t)} = 1$ (via Lagrangian)

(i) $\nabla_{\pi_k^{(t)}}\left(\mathcal{L}(\theta, q) - \lambda\left(\sum_{j=1}^K \pi_j^{(t)} - 1\right)\right) = \sum_{i=1}^n \frac{Q_{ik}}{\pi_k^{(t)}} - \lambda = 0$

$\implies \pi_k^{(t)} = \frac{\sum_{i=1}^n Q_{ik}}{n} \quad \because \sum_{k=1}^K \sum_{i=1}^n Q_{ik} = \sum_{i=1}^n \underbrace{\sum_{k=1}^K Q_{ik}}_{=1} = n$

(ii) $\nabla_{\lambda_k^{(t)}} \mathcal{L}(\theta, q) = \sum_{i=1}^n Q_{ik} \nabla_{\lambda_k^{(t)}} \log f_k\left(x_i; \lambda_k^{(t)}\right) = 0$

$\implies \mu_k^{(t)} = \frac{\sum_{i=1}^n Q_{ik}^{(t)} x_i}{\sum_{i=1}^n Q_{ik}^{(t)}}$, in mixture normal as $\log f_k\left(x_i; \lambda_k^{(t)} = \mu_k^{(t)}\right) \propto -\frac{1}{2\sigma^2}\left(x_i - \mu_k^{(t)}\right)^2$

.

$t = t + 1$

end

Return the responsibilities $\{Q_{ik}^{(t)}\}$ and the parameters $\mu_{1:K}^{(t)}, \pi_{1:K}^{(t)}$.

• **Generalized EM algorithm**: when updates of the parameters in the M-step is not exact, use gradient ascent: $\lambda_k^{(r+1)} = \lambda_k^{(r)} + \alpha \sum_{i=1}^n Q_{ik} \nabla_{\lambda_k} \log f_k\left(x_i; \lambda_k^{(r)}\right)$

3.8 Variational Inference

3.8.1 ELBO and Variational EM

Evidence Lower Bound (ELBO): For any (tractable) variational distribution $q(\mathbf{Z}, \theta) \in \mathcal{Q}$ parametrized by ϕ (**variational distribution**),

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z}, \theta)] - \mathbb{E}_q[\log q(\mathbf{Z}, \theta)] \\ &= \log p(\mathbf{X}) + \mathbb{E}_q[\log p(\mathbf{Z}, \theta \mid \mathbf{X})] - \mathbb{E}_q[\log q(\mathbf{Z}, \theta)] \\ &= \log p(\mathbf{X}) - \mathrm{KL}(q(\mathbf{Z}, \theta) \| p(\mathbf{Z}, \theta \mid \mathbf{X})) \le \log p(\mathbf{X})\end{aligned}$$

• Note that this is a joint distribution over the latent and the parameter, which is different from that in LVM.

Variational EM: Assume the variational distribution factorizes $q(\mathbf{Z}, \theta) = q(\mathbf{Z})q(\theta)$. Hence can be optimized iteratively via coordinate ascent:

(i) Fix q_θ and solve for $q_{\mathbf{Z}}$ that maximizes ELBO, with $q_{\mathbf{Z}}(\mathbf{Z}) \propto \exp\left(\int \log p(\mathbf{X}, \mathbf{Z}, \theta) q_\theta(\theta) d\theta\right)$;
(ii) Fix $q_{\mathbf{Z}}$ and solve for q_θ that maximizes ELBO, with $q_\theta(\theta) \propto \exp\left(\int \log p(\mathbf{X}, \mathbf{Z}, \theta) q_{\mathbf{Z}}(\mathbf{Z}) d\mathbf{Z}\right)$.
• In Bayesian approach, $\mathbf{Z}$ and θ can be treated equally due to the symmetry.

Mean-field approximation: Assume fully factorized variational distribution $q(\mathbf{Z}) = \prod_{j=1}^m q_j(z_j)$, with all θ above are also treated as $\mathbf{Z}$.

Coordinate Ascent Variational Inference (CAVI): related to **Gibbs Sampling** and iteratively approximating the full conditionals $p(z_j|\mathbf{Z}_{-j}, \mathbf{X}) \propto p(\mathbf{X}, \mathbf{Z})$, where $\mathbf{Z}_{-j} = [z_1, \cdots, z_{j-1}, z_{j+1}, \cdots, z_n]$.

Algorithm 17: Coordinate Ascent Variational Inference (CAVI)

while *ELBO not converge* **do**
  for $j = 1, \cdots, m$ **do**

$$q_j(z_j) \propto \exp\left(\mathbb{E}_{\mathbf{Z}_{-j} \sim q}\left[\log p(z_j \mid \mathbf{Z}_{-j}, \mathbf{X})\right]\right),$$

$$\text{if } p(z_j|\mathbf{Z}_{-j}, \mathbf{X}) \in \text{Exp-fam} \implies p(z_j|\mathbf{Z}_{-j}, \mathbf{X}) = h(z_j) \exp\left[z_j^\top \eta_j(\mathbf{Z}_{-j}, \mathbf{X}) - A(\eta_j(\mathbf{Z}_{-j}, \mathbf{X}))\right]$$

$$= \exp\left(\log h(z_j) + z_j^\top \mathbb{E}_{\mathbf{Z}_{-j}}\left[\eta_j(\mathbf{Z}_{-j}, \mathbf{X})\right] - \mathbb{E}_{\mathbf{Z}_{-j}}\left[A(\eta_j(\mathbf{Z}_{-j}, \mathbf{X}))\right]\right)$$

$$\propto h(z_j) \exp\left(z_j^\top \mathbb{E}_{\mathbf{Z}_{-j}}\left[\eta_j(\mathbf{Z}_{-j}, \mathbf{X})\right]\right)$$

  end
end
Return $q(\mathbf{Z}) = \prod_{j=1}^m q_j(z_j)$.

3.8.2 Variational Auto-Encoder (VAE)

Contains an **inference network (encoder)** $q_\phi(z|x)$ and a **generative network (decoder)** $p_\theta(x|z)$, where θ, ϕ are trained by simultaneously optimizing the ELBO with respect to both (via stochastic gradient approach), under the factorization assumption.

Amortised Inference: Joint with the prior $p(z)$ (often assume to be standard Gaussian), the decoder gives the generative model: $p(z)p_\theta(x|z)$, whereas the encoder can be thought of as a variational approximator of the posterior $q_\phi(z|x) \sim p_\theta(z|x)$.
• Amortised Inference uses a function approximator which is learnt at 'training time,' so that inference for new data items at 'test time' (i.e. when dealing with a particular datapoint or dataset depending on context) can be performed efficiently using the function approximator.

ELBO in VAE:

$$\begin{aligned}\mathcal{L}(\mathbf{X},\theta,\phi) &:= \sum_{i=1}^{n}\mathcal{L}\left(x_i,\theta,\phi\right) = \sum_{i=1}^{n}\mathbb{E}_{q_\phi(z_i|x_i)}\left[\log\frac{p_\theta\left(x_i,z_i\right)}{q_\phi\left(z_i\mid x_i\right)}\right] \leq \log p_\theta(\mathbf{X}) = \log\mathbb{E}_{p(\mathbf{Z})}\left[p_\theta(\mathbf{X}\mid\mathbf{Z})\right] \\ &= \sum_{i=1}^{n}\log p_\theta(x_i) - \mathbb{D}_{\mathrm{KL}}\left(q_\phi(z_i\mid x_i)\|p_\theta(z_i\mid x_i)\right) \\ &= \sum_{i=1}^{n}\mathbb{E}_{q_\phi(z_i|x_i)}\left[\log\frac{p_\theta(x_i,z_i)}{q_\phi(z_i\mid x_i)}\right] \\ &= \sum_{i=1}^{n}\mathbb{E}_{q_\phi(z_i|x_i)}\left[\log p_\theta(x_i\mid z_i)\right] + \mathbb{E}_{q_\phi(z_i|x_i)}\left[\log\frac{p(z_i)}{q_\phi(z_i\mid x_i)}\right] \\ &= \sum_{i=1}^{n}\underbrace{\mathbb{E}_{q_\phi(z_i|x_i)}\left[\log p_\theta(x_i\mid z_i)\right]}_{\text{reconstruction loss}} - \underbrace{\mathbb{D}_{\mathrm{KL}}\left(q_\phi(z_i\mid x_i)\|p(z_i)\right)}_{\text{regularizer}}\end{aligned}$$

• Reconstruction loss measures how effectively the original input is preserved when passed through the encoder and then back through the decoder.
• Regularizer increases the entropy of the encoding process to enforce smoothness in the $z-$space.
• Under Gaussian assumption of the encoder/decoder:$p_\theta(x\mid z) = \mathcal{N}\left(x;\mu_\theta(z),\sigma^2_\theta(z)\right) \quad q_\phi(x\mid z) = \mathcal{N}\left(z;\mu_\phi(x),\sigma^2_{phi}(x)\right)$, regularizer has the closed form:

$$\begin{aligned}\mathbb{D}_{\mathrm{KL}}\left(q_\phi(z\mid x)||p(z)\right) &= \frac{1}{2}\left[\mu_\phi(x)^\top\mu_\phi(x) + \mathrm{tr}\left(\Sigma_\phi(x)\right) - \log\det\left(\Sigma_\phi(x)\right) - k\right] \\ &= \frac{1}{2}\sum_{i=1}^{k}\left[\mu^2_{\phi,j}(x) + \sigma^2_{\phi,j}(x) - \log\left(\sigma^2_{\phi,j}(x)\right) - 1\right]\end{aligned}$$

Unbiased Estimator of ELBO:
• For θ is simply drawing $|B|$ samples from the joint $p_{data}(x)q_\phi(z|x)$, where p_{data} is the empirical data distribution.
• For ϕ is hard because it affects the expectation, hence consider the following 2 tricks:
(i) **Reparametrization**: $\nabla_\phi\mathbb{E}_{q_\phi(z_i|x_i)}\left[\log\left(\frac{p_\theta(x_i,z_i)}{q_\phi(z_i|x_i)}\right)\right] = \mathbb{E}_{q(\epsilon)}\left[\nabla_\phi\log\left(\frac{p_\theta(x_i,g(\epsilon_i,x_i,\phi))}{q_\phi(g(\epsilon_i,x_i,\phi)|x_i)}\right)\right]$
where $\epsilon\sim q(\epsilon) = N(0,I)$ and $z = g(\epsilon,x,\phi)$ (note now z depends on ϕ).
- e.g. To draw $z_i\sim N(z;\mu_\phi(x),\sigma^2_\phi(x))$ can be done via $z_i = \mu_\phi(x) + \sigma^2_\phi(x)\epsilon_i, \epsilon_i\sim N(0,I)$.
(ii) **Score function gradient (REINFORCE gradient)**:

$$\begin{aligned}\nabla_\phi\mathcal{L} &= \nabla_\phi\int q_\phi(z\mid x)\log\frac{p_\theta(x,z)}{q_\phi(z\mid x)}dz \\ &= \int\left(\nabla_\phi q_\phi(z\mid x)\right)\log\frac{p_\theta(x,z)}{q_\phi(z\mid x)}dz + \underbrace{\int q_\phi(z\mid x)\left(\nabla_\phi\log\frac{p_\theta(x,z)}{q_\phi(z\mid x)}\right)dz}_{=0,\text{ by score identity}} \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\left(\nabla_\phi\log q_\phi(z\mid x)\right)\log\frac{p_\theta(x,z)}{q_\phi(z\mid x)}\right]\end{aligned}$$

- **score identity**: $\mathbb{E}_{q_\phi(z)}\left[\nabla_\phi\log q_\phi(z)\right] = \int q_\phi(z)\nabla_\phi\log q_\phi(z)dz = \int\nabla_\phi q_\phi(z)dz = \nabla_\phi\int q_\phi(z)dz = 0$

Variations of the Lower Bounds: Any inference method that produces an unbiased marginal likelihood estimator will produce a valid lower bound.

(i) **Importance sampling**: independently sampling from the encoder K times $z_i \overset{iid}{\sim} q_\phi(\cdot|x)$

$$\mathcal{L}_K(x, \theta, \phi) = \mathbb{E}_{\prod_{k=1}^K q_\phi(z_k|x)} \left[\log \left(\frac{1}{K} \sum_{k=1}^K \frac{p_\theta(x, z_k)}{q_\phi(z_k \mid x)} \right) \right]$$

(ii) $\beta-$**ELBO**:

$$\mathcal{L}_\beta(x, \theta, \phi) = \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x \mid z)] - \beta \mathbb{D}_{\mathrm{KL}} (q_\phi(z \mid x) \| p(z))$$

• higher $\beta \implies$ larger regularizer $\implies$ smoother latent space (i.e. models were small changes in z lead to small changes in $p_\theta(x|z)$).