SC4/SM8 Advanced Topics in Statistical Machine Learning

Department of Statistics, University of Oxford, Hilary Term 2019
http://www.stats.ox.ac.uk/%7Esejdinov/atsml19/

Lecturer: Dino Sejdinovic

11th February 2019

1 Review of Fundamentals

1.1 Unsupervised Learning Basics

Notational remarks

- We will typically assume that we have collected p variables (features/attributes/dimensions) on n examples (items/observations) which can be represented as an $n \times p$ *data matrix* $\mathbf{X} = (x_{ij})$, where x_{ij} is the observed value of the j-th variable for the i-th example:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{np} \end{bmatrix}. \tag{1.1}$$

- We will denote the *rows* of $\mathbf{X}$ as $x_i \in \mathbb{R}^p$ and treat them as *column vectors*: i.e., the i-th item/example/observation x_i is the transpose of the i-th row of the data matrix $\mathbf{X}$:

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix} = [x_{i1}, x_{i2}, \dots, x_{ip}]^\top, \quad i = 1, \dots, n. \tag{1.2}$$

- We often assume that $x_1, \dots, x_n$ are *independent and identically distributed (i.i.d.)* samples of a *random vector* X over $\mathbb{R}^p$. When referring to the j-th dimension of random vector X, we will write $X^{(j)}$.

Unsupervised learning is a broad and arguably more challenging part of machine learning. The goal of unsupervised learning is to extract key features of the "unlabelled" dataset. While in supervised learning our data items $\{x_i\}_{i=1}^n$ come with an extra piece of information which we are trying to predict, in unsupervised learning we are trying to understand the process which generated data $\{x_i\}_{i=1}^n$ itself. We will here review two basic unsupervised learning tasks: *dimensionality reduction* and *clustering*. Broadly speaking, dimensionality reduction aims to, for each data item $x_i \in \mathbb{R}^p$, find a lower dimensional representation $z_i \in \mathbb{R}^k$ with $k \ll p$ such that the map $x \mapsto z$ preserves certain *interesting statistical properties* in data. Clustering on the other hand, partitions the set of n data items into K disjoint groups. We will also review two instances of simple algorithms for each: *Principal Components Analysis (PCA)* for dimensionality reduction and *k-means algorithm* for clustering.

1.1.1 Dimensionality Reduction with PCA

Principal Components Analysis (PCA) is a dimensionality reduction technique which aims to preserve *variance* in the data. PCA is a *linear* dimensionality reduction technique: it essentially looks for a *new basis* to represent a noisy dataset.

For simplicity, we will assume for PCA that our dataset is *centred*, i.e., that its average is $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = 0$. If not, we can always subtract it from each x_i (this is called *data centering*). Thus, we can write the *sample covariance matrix* S as

$$S = \widehat{\mathrm{Cov}}(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^\top = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^\top = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}. \tag{1.3}$$

Note that the matrix S is symmetric and positive semi-definite.

PCA recovers an orthonormal basis $v_1, v_2, \ldots, v_p$ in $\mathbb{R}^p$ – vectors v_i are *called principal components (PC) or loading vectors* – such that:

- The first principal component (PC) v_1 is the *direction of greatest variance* of data.
- The j-th PC v_j is the *direction orthogonal to* $v_1, v_2, \ldots, v_{j-1}$ *of greatest variance*, for $j = 2, \ldots, p$.

Given this basis, the k-dimensional representation of data item x_i is the vector of projections of x_i onto the first k PCs:

$$z_i = V_{1:k}^\top x_i = \left[v_1^\top x_i, \ldots, v_k^\top x_i\right]^\top \in \mathbb{R}^k,$$

where $V_{1:k} = [v_1, \ldots, v_k]$ is a $p \times k$ matrix. This gives us the *transformed data matrix*, also called the *scores matrix*

$$\mathbf{Z} = \mathbf{X}V_{1:k} \in \mathbb{R}^{n\times k}. \tag{1.4}$$

Deriving the first principal component

Recall that we model our dataset is an i.i.d. sample $\{x_i\}_{i=1}^n$ of a random vector $X = \left[X^{(1)} \ldots X^{(p)}\right]^\top$. Projections to PCs define a linear transformation of X given by $Z = V_{1:k}^\top X$ which is a k-dimensional random vector. Dimensions of Z are called *derived variables*. Consider the first dimension of Z:

$$Z^{(1)} = v_1^\top X = v_{11}X^{(1)} + v_{12}X^{(2)} + \cdots + v_{1p}X^{(p)}. \tag{1.5}$$

The first PC $v_1 = [v_{11}, \ldots, v_{1p}]^\top \in \mathbb{R}^p$ is chosen to maximise the sample variance $\widehat{\mathrm{Var}}(Z^{(1)}) = v_1^\top \widehat{\mathrm{Cov}}(X) v_1$, i.e. it is defined as the solution to

$$\max_{v_1} \; v_1^\top S v_1$$
$$\text{subject to: } v_1^\top v_1 = 1.$$

By considering the Lagrangian:

$$\mathcal{L}(v_1, \lambda_1) = v_1^\top S v_1 - \lambda_1 \left(v_1^\top v_1 - 1 \right) \tag{1.6}$$

and the corresponding vector of partial derivatives

$$\frac{\partial \mathcal{L}(v_1, \lambda_1)}{\partial v_1} = 2Sv_1 - 2\lambda_1 v_1 \tag{1.7}$$

we obtain the eigenvector equation $Sv_1 = \lambda_1 v_1$, i.e. v_1 must be an eigenvector of S and the dual variable λ_1 is the corresponding eigenvalue. Since $v_1^\top S v_1 = \lambda_1 v_1^\top v_1 = \lambda_1$, the first PC must be the eigenvector associated with the *largest eigenvalue* of S.

Subsequent principal components

Similarly, the second PC maximizes the sample variance $\widehat{\mathrm{Var}}(Z^{(2)}) = v_2^\top \widehat{\mathrm{Cov}}(X) v_2$ of the second derived variable among the directions orthogonal to v_2, i.e.

$$\begin{aligned} &\max_{v_2} \; v_2^\top S v_2 \\ &\text{subject to: } v_2^\top v_2 = 1, \; v_1^\top v_2 = 0. \end{aligned}$$

Lagrangian is

$$\mathcal{L}(v_2, \lambda_2, \gamma_2) = v_2^\top S v_2 - \lambda_2 \left(v_2^\top v_2 - 1 \right) - \gamma_2 v_1^\top v_2 \tag{1.8}$$

and setting the corresponding vector of partial derivatives to zero

$$\frac{\partial \mathcal{L}(v_2, \lambda_2, \gamma_2)}{\partial v_2} = 2Sv_2 - 2\lambda_2 v_2 - \gamma_2 v_1 = 0. \tag{1.9}$$

Left-multiplying (1.9) by $v_1^\top$ gives $2v_1^\top S v_2 = \gamma_2$. However, since S is symmetric and v_1 is its eigenvector, we have

$$\gamma_2 = 2v_1^\top S v_2 = 2v_2^\top S v_1 = 2\lambda_1 v_2^\top v_1 = 0. \tag{1.10}$$

Hence $Sv_2 = \lambda_2 v_2$ and similarly as before v_2 must be the eigenvector corresponding to the second largest eigenvalue λ_2 of S.

Continuing the process further, we obtain the *eigenvalue decomposition* of S given by

$$S = V \Lambda V^\top \tag{1.11}$$

where Λ is a diagonal matrix with eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0 \tag{1.12}$$

on the diagonal and V is a $p \times p$ orthogonal matrix (i.e. $VV^\top = V^\top V = I$) whose *columns* are the p eigenvectors of S, i.e. the principal components $v_1, \ldots, v_p$.

In summary,

- Derived scalar variable (projection to the j-th principal component) $Z^{(j)} = v_j^\top X$ has sample variance λ_j, for $j = 1, \ldots, p$.
- Derived variables are *uncorrelated*: $\mathrm{Cov}(Z^{(i)}, Z^{(j)}) \approx v_i^\top S v_j = \lambda_j v_i^\top v_j = 0$, for $i \neq j$.
- The *total sample variance* is given by $\mathrm{Tr}(S) = \sum_{i=1}^p S_{ii} = \lambda_1 + \ldots + \lambda_p$, so the *proportion of total variance explained* by the j^{th} PC is $\frac{\lambda_j}{\lambda_1+\lambda_2+\ldots+\lambda_p}$

Reconstruction view of PCA

We can map back to the original p-dimensional space using

$$\hat{x}_i = V_{1:k} V_{1:k}^\top x_i. \tag{1.13}$$

This is a *reconstruction* of data item x_i. It can be shown (problem sheet) that PCA gives the *optimal linear reconstruction* based on a k-dimensional compression.

PCA via the Singular Value Decomposition

PCA can also be understood using the Singular Value Decomposition (SVD) of data matrix $\mathbf{X}$. Recall that any real-valued $n \times p$ matrix $\mathbf{X}$ can be written as $\mathbf{X} = UDV^\top$ where

- U is an $n \times n$ orthogonal matrix: $UU^\top = U^\top U = I_n$.
- D is a $n \times p$ matrix with decreasing *non-negative* elements on the diagonal (the singular values of $\mathbf{X}$) and zero off-diagonal elements.
- V is a $p \times p$ orthogonal matrix: $VV^\top = V^\top V = I_p$.

Note that

$$(n-1)S = \mathbf{X}^\top \mathbf{X} = (UDV^\top)^\top (UDV^\top) = VD^\top U^\top UDV^\top = VD^\top DV^\top,$$

using orthogonality of U. The eigenvalues of S are thus the diagonal entries of $\Lambda = \frac{1}{n-1} D^\top D$.

We also have

$$\mathbf{X}\mathbf{X}^\top = (UDV^\top)(UDV^\top)^\top = UDV^\top VD^\top U^\top = UDD^\top U^\top,$$

using orthogonality of V.

The $n \times n$ matrix $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ with entries $\mathbf{K}_{ij} = x_i^\top x_j$ is called the *Gram matrix* of dataset $\mathbf{X}$. Note that $\mathbf{K}$ and $(n-1)S = \mathbf{X}^\top \mathbf{X}$ have the same nonzero eigenvalues, equal to the non-zero squared singular values of $\mathbf{X}$ (non-zero entries on the diagonals of $D^\top D$ and $DD^\top$).

If we consider projections to *all principal components*, the transformed data matrix is

$$\mathbf{Z} = \mathbf{X}V = UDV^\top V = UD, \tag{1.14}$$

If $p \leq n$ this means

$$z_i = \left[U_{i1}D_{11}, \ldots, U_{ip}D_{pp}\right]^\top, \tag{1.15}$$

and if $p > n$ only the first n projections are defined (sample covariance will be at most rank n):

$$z_i = \left[U_{i1}D_{11}, \ldots, U_{in}D_{nn}, 0, \ldots, 0\right]^\top. \tag{1.16}$$

Thus, $\mathbf{Z}$ can be obtained from the eigendecomposition of Gram matrix $\mathbf{K}$. When $p \gg n$, eigendecomposition of $\mathbf{K}$ requires much less computation, $O(n^3)$, than the eigendecomposition of the covariance matrix, $O(p^3)$, so is the preferred method for PCA in that case.

1.1.2 Clustering

Clustering is one of the fundamental and ubiquitous tasks in exploratory data analysis – a first intuition about the data is often based on identifying meaningful disjoint groups among the data items. In *partition-based* clustering, which we consider in this note, one divides n data items into K clusters $C_1, \ldots, C_K$ where for all $k, k' \in \{1, \ldots, K\}$,

$$C_k \subset \{1, \ldots, n\}, \qquad C_k \cap C_{k'} = \emptyset \;\; \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}.$$

Central to the goals of clustering is the notion of similarity/dissimilarity between data items. There will be many ways to define the notion of similarity, and the choice will depend on the dataset being analyzed and dictated by domain specific knowledge.

Intuitively, clustering aims to group similar items together and to place separate dissimilar items into different groups. However, note that these two objectives in many cases contradict each other (similarity is not a transitive relation, while being in the same cluster is an equivalence relation). One could imagine a long sequence of items such that each next item is very similar to the previous one so that they should all belong to the same cluster – but that would also mean that the endpoints are potentially highly dissimilar. Hence, there are also different clustering techniques which emphasize different aspects of these goals, i.e. whether to keep similar points together or dissimilar points apart.

There have been several attempts to construct an axiomatic definition of clustering, but it is surprisingly difficult to put on rigorous footing. Consider the following three basic properties required of a clustering method $\mathcal{F} : (\mathcal{D} = \{x_i\}_{i=1}^n, \rho) \mapsto \{C_1, \ldots, C_K\}$ which takes as an input dataset $\mathcal{D}$ and a dissimilarity function ρ and returns a partition of $\mathcal{D}$:

- **Scale invariance.** For any $\alpha > 0$, $\mathcal{F}(\mathcal{D}, \alpha\rho) = \mathcal{F}(\mathcal{D}, \rho)$, i.e. partition should not depend on units in which dissimilarity is measured.

- **Richness.** For any partition $C = \{C_1, \ldots, C_K\}$ of $\mathcal{D}$, there exists dissimilarity ρ, such that $\mathcal{F}(\mathcal{D}, \rho) = C$, i.e. the outcome is fully controlled by the dissimilarity function.

- **Consistency.** If ρ and ρ' are two dissimilarities such that for all $x_i, x_j \in \mathcal{D}$ the following holds:

$$\begin{aligned} x_i, x_j \text{ belong to the same cluster in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \leq \rho(x_i, x_j) \\ x_i, x_j \text{ belong to different clusters in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \geq \rho(x_i, x_j), \end{aligned}$$

  then $\mathcal{F}(\mathcal{D}, \rho') = \mathcal{F}(\mathcal{D}, \rho)$. In other words, if the items in the same cluster become more similar and the items already separated become less similar, then the clustering should not change.

While all three properties appear natural, Kleinberg's impossibility theorem [15] states that there exists no clustering method that satisfies all three properties, implying that every clustering method will have some undesirable properties. For further discussion, see Section 22.5 in [25].

We will consider here the simplest widely used clustering method: K-means algorithm (and its extension, DP-means).

K-means algorithm

K-means is the simplest partition-based clustering algorithm. It uses a preassigned number of clusters and represents each cluster using a *prototype* or *cluster centroid* μ_k.

The idea of K-means is to measure the quality of each cluster using its *within-cluster deviance* from the cluster centroids

$$W(C_k, \mu_k) = \sum_{i \in C_k} \|x_i - \mu_k\|_2^2.$$

The overall quality of the clustering is then given by the total within-cluster deviance:

$$W(\{C_k\}, \{\mu_k\}) = \sum_{k=1}^{K} W(C_k, \mu_k) = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|_2^2,$$

where $c_i = k$ if and only if $i \in C_k$. This is now the overall objective function used to select both the cluster centroids and the assignment of points to clusters. The joint optimization over both the partition $\{C_k\}$ and centroids $\{\mu_k\}$ is a combinatorial optimization problem and is computationally hard. However, note that

- Given partition $\{C_k\}$, we can easily find the optimal centroids by differentiating W with respect to μ_k:

$$\frac{\partial W}{\partial \mu_k} = 2 \sum_{i \in C_k} (x_i - \mu_k) = 0 \qquad \Rightarrow \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

Algorithm 1.1 K-means algorithm

Input: dataset $\mathcal{D} = \{x_i\}_{i=1}^n$, desired number of clusters K
Output: partition $\{C_1, \ldots, C_K\}$

Randomly initialize K cluster centroids $\mu_1, \ldots, \mu_K$.
while the partition has not converged **do**

- *Cluster assignment:* For each $i = 1, \ldots, n$, assign each x_i to the cluster with the nearest centroid,

$$c_i := \text{argmin}_{k=1,\ldots,K} \|x_i - \mu_k\|_2^2$$

  Set $C_k := \{i : c_i = k\}$ for each k.

- *Move centroids:* Set $\mu_1, \ldots, \mu_K$ to the averages of the new clusters:

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

return partition $\{C_1, \ldots, C_K\}$

- Given prototypes, we can easily find the optimal partition by assigning each data point to the closest cluster prototype:

$$c_i = \text{argmin}_k \|x_i - \mu_k\|_2^2 .$$

Thus one can employ an iterative alternating optimization, which is exactly the K-means algorithm:

K-means is a heuristic search algorithm so it can (and often will) get stuck at local optima. The result depends on the starting configurations. Typically one performs a number of runs from different random initial values of centroids, and then chooses the end result with minimum W. Since each step does not increase the objective function and the number of possible partitions is finite, the algorithm will converge to a local optimum. However, note that there could be ties in the cluster assignment, which need to be broken in a systematic fashion.

K-means++

A simple yet provably effective solution to the problem of initialization of centroids in the K-means algorithm was proposed by [1]. The method starts with sampling a data item from $\mathcal{D} = \{x_i\}_{i=1}^n$ uniformly at random and making it centroid μ_1. We then compute the squared distances $\rho_i^2 = \|x_i - \mu_1\|_2^2$. Centroid μ_2 is then initialized to another data item sampled using the probability mass function $p(i) = \rho_i^2 / \sum_{j=1}^n \rho_j^2$ and the process continues with the probability mass function being updated at each step, i.e. to initialize

k-th centroid μ_k, we compute

$$\rho_i^2 = \min\left\{\|x_i - \mu_1\|_2^2, \ldots, \|x_i - \mu_{k-1}\|_2^2\right\}. \tag{1.17}$$

Remarkably, this method comes with a precise theoretical guarantee. In particular, [1] show that if clustering $\{C_k^{++}\}$ is obtained using K-means++ then

$$\mathbb{E}\left[W\left(\{C_k^{++}\}\right)\right] \leq 8\left(\log K + 2\right) W^*, \tag{1.18}$$

where W^* is the within-cluster deviance of the globally optimal clustering and the expectation is taken over the random sampling used in the initialisation.

DP-means

K-means is intuitive and straightforward to implement, but how do we select the number of clusters K in the first place? Clearly, the objective function is minimized (and equals zero) if we let $K = n$, but this is not a meaningful clustering.

One elegant approach is the *DP-means algorithm* [16] that comes from the interpretation of K-means using small variance asymptotics of the Expectation Maximization (EM) algorithm for mixture modelling. We will discuss mixture modelling and EM algorithm later in the course. DP-means starts from a single cluster, i.e. $K = 1$ and modifies the cluster assignment step as follows:

1. Initialize $K = 1$ and $\mu_1 = \frac{1}{n}\sum_{i=1}^n x_i$ (the global mean).
2. *DP-means cluster assignment:* For each $i = 1, \ldots, n$,
   - if $\min_{k=1,\ldots,K} \|x_i - \mu_k\|_2^2 > \lambda$, set $K \leftarrow K + 1$, $c_i \leftarrow K$, $\mu_K \leftarrow x_i$
   - otherwise, set $c_i = \operatorname{argmin}_{k=1,\ldots,K} \|x_i - \mu_k\|_2^2$.

The rest of the algorithm is exactly the same as K-means. Tuning parameter λ controls the tradeoff between the traditional K-means objective and the number of clusters. DP-means can be shown to locally minimize the objective

$$W_\lambda(\{C_k\}, \{\mu_k\}, K) = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 + \lambda K. \tag{1.19}$$

Indeed, just like in K-means algorithm, the "move centroids" step can only decrease the objective, whereas for every data item $i = 1, \ldots, n$, its assignment to the nearest centroid if closer than λ will not increase the objective and if the nearest centroid is at a distance larger than λ we can create another cluster and pay a penalty λ while still decreasing the overall objective (1.19).

1.2 Supervised Learning Basics

1.2.1 Loss and Risk

In the supervised learning framework, we are trying to learn a function $f : \mathcal{X} \to \mathcal{Y}$ from an input space $\mathcal{X}$ into an output space $\mathcal{Y}$ based on a set of paired examples $(x_1, y_1), \dots (x_n, y_n)$ and a given *loss function* L measuring discrepancy between predicted output values $f(x_i)$ and the true output values y_i at the inputs x_i. It is assumed that examples $(x_1, y_1), \dots (x_n, y_n)$ are i.i.d. samples from an *unknown joint probability distribution* $P_{X,Y}$ on $\mathcal{X} \times \mathcal{Y}$ and the goal of learning is to find the function f which *minimizes the expectation of the loss* over $P_{X,Y}$ - which is called *risk*.

Empirical Risk Minimisation (ERM) *Loss* is any function

$$L : \mathcal{Y} \times \mathcal{Y} \times \mathcal{X} \to \mathbb{R}^+. \tag{1.20}$$

Risk of a function $f : \mathcal{X} \to \mathcal{Y}$ is the expected loss:

$$R(f) = \mathbb{E}_{X,Y} L(Y, f(X), X). \tag{1.21}$$

For a given dataset $(x_1, y_1), \dots (x_n, y_n)$, the *empirical risk* of f is given by

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i), x_i). \tag{1.22}$$

The *Empirical Risk Minimisation* is the problem

$$\hat{f} = \operatorname{argmin}_{f \in \mathcal{H}} \hat{R}(f),$$

where $\mathcal{H}$ is a given class of functions (*hypothesis class*).

Remark 1. The ultimate goal of learning is to minimise the true risk - *not* the empirical risk, which is only an estimate of the true risk. But the true risk of any given function is unknown because the distribution $P_{X,Y}$ is unknown.

Remark 2. Loss function typically depend on the input x only through $f(x)$, so that with some abuse of notation we often write $L(y, f(x))$ instead of $L(y, f(x), x)$. $L(y, f(x))$ is usually some notion of distance between the true output y and the predicted output $f(x)$.

Examples of hypothesis classes. Hypothesis classes can be very simple, e.g. for $\mathcal{X} = \mathbb{R}^p$, we can consider all linear functions $f(x) = w^\top x + b$, parametrized by $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$, or we could consider a specific *nonlinear feature expansion* $\varphi : \mathcal{X} \to \mathbb{R}^D$, and a model linear in those features: $f(x) = w^\top \varphi(x) + b$, but nonlinear in the original inputs $\mathcal{X}$, parametrized by $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$. For example, starting with $\mathcal{X} = \mathbb{R}^2$, we can consider $\varphi\left(\begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}\right) = [x_{i1}, x_{i2}, x_{i1}^2, \sqrt{2} x_{i1} x_{i2}, x_{i2}^2]^\top$, such that the resulting function can

depend on quadratic and interaction terms as well. An important type of hypothesis classes we will consider in this course are *Reproducing Kernel Hilbert Spaces (RKHS)*, which are also linear in certain feature expansions but those feature expansions could potentially be infinite-dimensional.

Examples of loss functions. Loss functions come in many different forms. One of the main considerations for selecting loss functions is the type of outputs we are trying to predict, i.e., whether it is real-valued or discrete/categorical. Note that even if outputs are discrete, the function $f(x)$ we are trying to learn is typically real-valued. For example, in binary classification, the common convention is that the two classes are denoted by -1 and $+1$. One associates predictions of these classes with $\text{sign}(f(x))$, whereas the magnitude of $f(x)$ can be thought of as the confidence in those predictions (not necessarily in a probabilistic sense). The loss can penalize misclassification (wrong sign) as well as the overconfident misclassification (wrong sign and large magnitude) and even underconfident correct classification (correct sign but small magnitude). Thus, the loss functions can often be expressed as a function of $yf(x)$.

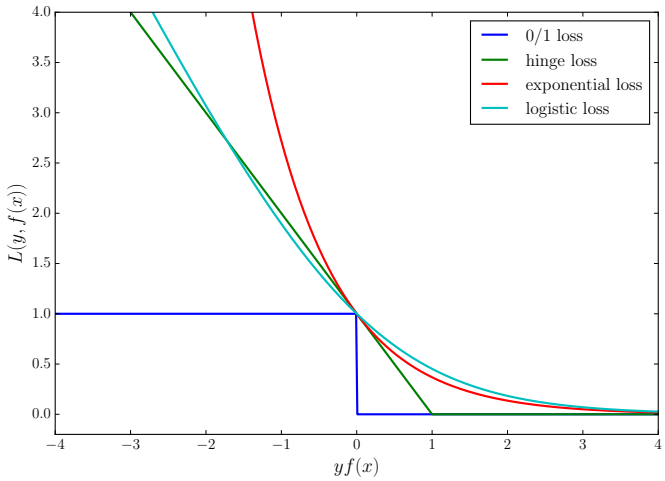

Figure 1.1: Loss functions for binary classification

Below are some loss functions commonly used in binary classification and regression.

- Binary classification:
  - 0/1 loss $L(y, f(x)) = \mathbf{1}\{yf(x) \leq 0\}$,
    (also called misclassification loss, optimal solution is called the *Bayes classifier* and is given by $f(x) = \text{argmax}_{k \in \{0,1\}} \mathbb{P}(Y = k | X = x)$),
  - hinge loss $L(y, f(x)) = (1 - yf(x))_+$
    (used in *support vector machines* - leads to sparse solutions),

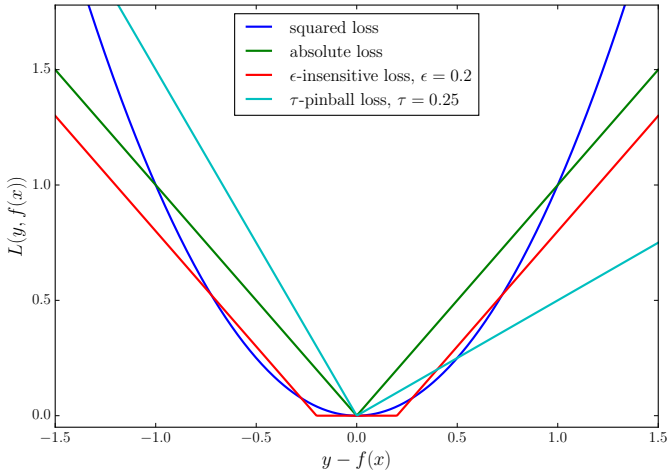

Figure 1.2: Loss functions for regression

  - exponential loss $L(y, f(x)) = e^{-yf(x)}$
    (used in *boosting* algorithms - Adaboost),
  - logistic loss $L(y, f(x)) = \log\left(1 + e^{-yf(x)}\right)$
    (used in *logistic regression*, and associated with a linear log-odds probabilistic model).

- Regression:
  - squared loss: $L(y, f(x)) = (y - f(x))^2$
    (*least squares regression*: optimal f is the conditional mean $\mathbb{E}[Y|X = x]$),
  - absolute loss: $L(y, f(x)) = |y - f(x)|$
    (*least absolute deviations regression*, which is less sensitive to outliers: optimal f is the conditional median $\text{med}[Y|X = x]$),
  - τ-pinball loss: $L(y, f(x)) = 2\max\{\tau(y - f(x)), (\tau - 1)(y - f(x))\}$ for $\tau \in (0, 1)$
    (*quantile regression*: optimal f is the τ-quantile of $p(y|X = x)$),
  - ϵ-insensitive (Vapnik) loss: $L(y, f(x)) = \begin{cases} 0, \text{ if } |y - f(x)| \leq \epsilon, \\ |y - f(x)| - \epsilon, \text{ otherwise.} \end{cases}$
    (*support vector regression*, which leads to sparse solutions).

In binary classification, 0/1 is an idealised version of loss which penalizes misclassification regardless of the magnitude of $f(x)$. However, ERM under 0/1 loss is NP hard[1]. Therefore, we typically use *convex upper bound surrogate losses* (hinge, exponential, lo-

[1]It is NP-hard to even approximately minimize the ERM under 0/1 loss - i.e. there is no known polynomial-time algorithm to obtain a solution which is a small constant worse than the optimum.

gistic[2]). What is the importance of the convexity of loss as a function of $yf(x)$ as shown in Fig. 1.1? Consider the hypothesis class $f(x) = w^\top \varphi(x)$, with $w \in \mathbb{R}^D$ (we ignore the intercept to simplify notation) and assume that $L(y, f(x)) = \rho(yf(x))$ for a convex differentiable function ρ. Then the empirical risk and its gradient are given by

$$\hat{R}(w) = \frac{1}{n}\sum_{i=1}^{n} \rho\left(y_i w^\top \varphi(x_i)\right), \quad \frac{\partial \hat{R}}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} \rho'\left(y_i w^\top \varphi(x_i)\right) y_i \varphi(x_i).$$

Furthermore, the Hessian matrix of the empirical risk is given by

$$\frac{\partial^2 \hat{R}}{\partial w \partial w^\top} = \frac{1}{n}\sum_{i=1}^{n} \rho''\left(y_i w^\top \varphi(x_i)\right) \varphi(x_i)\varphi(x_i)^\top, \tag{1.23}$$

using $y_i^2 = 1$. This Hessian is now a positive semidefinite matrix which can be seen from $\rho''(t) \geq 0\ \forall t$ and

$$\alpha^\top \frac{\partial^2 \hat{R}}{\partial w \partial w^\top} \alpha \quad = \quad \frac{1}{n}\sum_{i=1}^{n} \rho''\left(y_i w^\top \varphi(x_i)\right) \left(\alpha^\top \varphi(x_i)\right)^2 \geq 0.$$

for any $\alpha \in \mathbb{R}^D$. Thus, empirical risk is a convex function of w and thus has a *unique minimum*. Typically, there is no closed form solution for w and iterative optimisation techniques like *gradient ascent* or *Newton-Raphson algorithm* are used.

1.2.2 Regularisation

Recall that we are not ultimately interested in the exact minimizer of the *empirical risk* but in that of the *true risk*. ERM thus risks *overfitting*: when the hypothesis class is complex, one can easily find a function that matches the observed examples exactly but does not *generalise* to the new examples.

The idea behind *regularisation* is to limit the flexibility of hypothesis class in order to prevent overfitting. For the hypothesis space $\mathcal{H} = \{f_\theta : \theta \in \Theta\}$, this is achieved by adding the term which *penalises the large values of parameters* θ to the ERM criterion:

$$\min_\theta \hat{R}(f_\theta) + \lambda \|\theta\|_\rho^\rho = \min_\theta \frac{1}{n}\sum_{i=1}^{n} L(y_i, f_\theta(x_i)) + \lambda \|\theta\|_\rho^\rho$$

where $\rho \geq 1$, and $\|\theta\|_\rho = (\sum_{j=1}^{p} |\theta_j|^\rho)^{1/\rho}$ is the L_ρ norm of θ (also of interest when $\rho \in [0, 1)$, but this is no longer a norm). These methods are also known as *shrinkage* methods since their effect is to shrink parameter estimates towards 0. Note that we have an additional *tuning parameter* (or *hyperparameter*) λ which controls the amount of regularisation, and, as a result, also controls the complexity of the model.

[2] to make it into an upper bound on 0/1, divide the logistic loss by log(2) - rescaling of the loss does not change the ERM problem

The most common forms of regularisation include *Ridge regression / Tikhonov regularization*: $\rho = 2$, *LASSO* penalty: $\rho = 1$, and *elastic net* regularization with a mixed L_1/L_2 penalty:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(x_i)) + \lambda \left[(1-\alpha)\|\theta\|_2^2 + \alpha\|\theta\|_1 \right].$$

In some hypothesis classes, it is possible to directly penalise some notion of *smoothness* of the function f we are trying to learn, e.g. for $\mathcal{X} = \mathbb{R}$, the regularisation term can consist of the *Sobolev norm*

$$\|f\|_{W^1}^2 = \int_{-\infty}^{+\infty} f(x)^2 dx + \int_{-\infty}^{+\infty} f'(x)^2 dx, \tag{1.24}$$

which penalises functions with large derivative values.

1.2.3 Examples of ERM

Regularised Least Squares / Ridge Regression

This corresponds to the squared loss $L(y, f(x)) = (y - f(x))^2$. For linear functions $f(x) = w^\top x + b$, we have

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top x_i - b)^2 + \frac{\lambda}{n} \|w\|_2^2. \tag{1.25}$$

Note the rescaling of the regularisation term and that the bias term b is not included in the regularisation. This is important as otherwise the predictions would depend on the origin for the response variables y (i.e. adding a constant c to each target would result in different predictions from simply shifting the original predictions by c). Fortunately, when using centred inputs, i.e., $\sum_{i=1}^{n} x_i = 0$, b can be estimated by $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, so we can also assume that the responses are centred and remove the intercept from the model. We obtain the problem

$$\min_{w} \|\mathbf{y} - \mathbf{X}w\|_2^2 + \lambda \|w\|_2^2. \tag{1.26}$$

Differentiating and setting to zero gives the closed form solution

$$w = \left(\mathbf{X}^\top \mathbf{X} + \lambda I \right)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{1.27}$$

Logistic Regression

Despite the name, *logistic regression is a method for classification.* It uses the logistic loss $L(y, f(x)) = \log\left(1 + e^{-yf(x)}\right)$. Hence, again for a linear classifier $f(x) = w^\top x + b$,

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i(w^\top x_i + b)}\right) + \frac{\lambda}{n} \|w\|_2^2. \tag{1.28}$$

Logistic regression can also be associated to a probabilistic model. Namely, assume that the function of interest $f(x) = w^\top x + b$ models the log-odds ratio:

$$\log \frac{p(y_i = +1|w, b, x_i)}{p(y_i = -1|w, b, x_i)} = w^\top x_i + b. \tag{1.29}$$

Then the conditional distribution of $Y|X$ is given by

$$p(y_i = +1|w, b, x_i) = \frac{1}{1 + e^{-(w^\top x_i + b)}} = \sigma(w^\top x_i + b), \tag{1.30}$$

$$p(y_i = -1|w, b, x_i) = \frac{1}{1 + e^{w^\top x_i + b}} = \sigma(-w^\top x_i - b), \tag{1.31}$$

where we denoted by $\sigma(t) = 1/(1 + e^{-t})$ the *logistic function* which maps the real line to $(0, 1)$ interval. Note that the logistic function satisfies $\sigma(-t) = 1 - \sigma(t)$. Thus, we can write (1.30) and (1.31) as $p(y_i|w, b, x_i) = \sigma(y_i(w^\top x_i + b))$ and the conditional log-likelihood of the outputs given the inputs is

$$\log p(\mathbf{y}|w, b, \mathbf{X}) = \log \prod_{i=1}^{n} \sigma(y_i(w^\top x_i + b)) = -\sum_{i=1}^{n} \log \left(1 + e^{-y_i(w^\top x_i + b)}\right).$$

Thus finding the parameters w and b that maximise the conditional log-likelihood is equivalent to minimising the empirical risk corresponding to the logistic loss, which is the negative log-likelihood of the linear log-odds model. Moreover, the regularisation term can be interpreted as a normal prior on w in *Bayesian logistic regression*. Again, there is no closed form solution for logistic regression, but the objective is convex and differentiable and the numerical optimisation via gradient ascent or Newton-Raphson algorithm can be used.

The connection between maximisation of the log-likelihood and minimisation of the empirical risk extends beyond logistic regression. Indeed, in the context of classification, whenever $p(y_i|x_i, \theta)$ is a log-concave function of $y_i f_\theta(x_i)$, we can define a convex loss $\rho(y f_\theta(x)) = -\log p(y_i|x_i, \theta)$. But the converse is not true, e.g. hinge loss used in the SVMs below does not correspond to a negative log-likelihood in any probabilistic model (unless additional artificial classes are introduced).

Support Vector Machines

Support Vector Machines (SVMs) for classification use hinge loss, $L(y, f(x)) = \max\{0, 1 - yf(x)\}$. Thus, for a linear classifier $f(x) = w^\top x + b$, we obtain

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i(w^\top x_i + b)\} + \frac{\lambda}{n} \|w\|_2^2. \tag{1.32}$$

This does not have a closed form solution and requires numerical optimisation. Eq. (1.32) is not how you would typically see an SVM written in the literature, though. Rather, we introduce a substitution $\xi_i = \max\{0, 1 - y_i(w^\top x_i + b)\}$, which implies that

$\xi_i \geq 0$, $y_i(w^\top x_i + b) \geq 1 - \xi_i$ and with a reparametrisation of the regularisation parameter $C = 1/2\lambda$ obtain the following equivalent form, called C-SVM:

$$\begin{aligned} &\min_{w,b,\xi} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \right), \\ &\text{subject to} \quad \xi_i \geq 0, \qquad y_i \left(w^\top x_i + b \right) \geq 1 - \xi_i. \end{aligned} \tag{1.33}$$

SVMs have the following nice property: the normal vector w of the hyperplane determining the classification rule can be written as $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ where a large number of α-coefficients (so called *dual coefficients*) is typically zero. Thus, only a small number of datapoints (*support vectors*, those with a non-zero α) determine the learned classification rule. In the next chapter, we will make a deep dive into SVMs, introducing it from a completely different perspective, that of *maximum margin classification*.

2 Support Vector Machines

These notes are a revised version of lecture notes from the UCL course "Reproducing Kernel Hilbert Spaces in Machine Learning" [12], reproduced here by courtesy of Arthur Gretton.

2.1 Duality in Convex Optimization

We will need some basic results from duality in convex optimization in order to study support vector machines, one of the fundamental techniques for classification. This review covers the material from [6, Sections 5.1-5.5].

2.1.1 The Lagrangian

Consider a constrained optimization problem of an objective function $f_0 : \mathbb{R}^n \to \mathbb{R}$, with m inequality and r equality constraints:

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0 & i = 1, \ldots, m \\ & h_j(x) = 0 & j = 1, \ldots r. \end{array} \tag{2.1}$$

We denote $\mathcal{D} := \bigcap_{i=0}^{m} \text{dom} f_i \cap \bigcap_{j=1}^{r} \text{dom} h_j$, and require the domain $\mathcal{D} \subseteq \mathbb{R}^n$ where the objective function f_0 and the constraint functions $f_1, \ldots, f_m, h_1, \ldots, h_r$ are all defined to be nonempty. We will refer to (2.1) as the primal problem and denote by $p^* = f_0(x^*)$ its optimal value. Any point $\tilde{x} \in \mathcal{D}$ for which constraints are satisfied , i.e. $f_i(\tilde{x}) \leq 0$, $h_j(\tilde{x}) = 0$, is called a **primal feasible** point.

The **Lagrangian** $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}$ associated with problem (2.1) is given by

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

The vectors $\lambda \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^r$ are called **Lagrange multipliers** or **dual variables**. The **Lagrange dual function** (or just "dual function") is written

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu).$$

The domain of g, dom$g$, is the set of values (λ, μ) for which the Lagrangian is bounded from below, i.e. $g > -\infty$. The dual function is a pointwise infimum of affine[1] functions of (λ, ν), hence it is concave in (λ, ν) [6, p. 83]. A **dual feasible** pair (λ, ν) is a pair for which $\lambda \succeq 0$ and $(\lambda, \nu) \in \text{dom} g$.

[1] A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is affine if it takes the form $f(x) = Ax + b$.

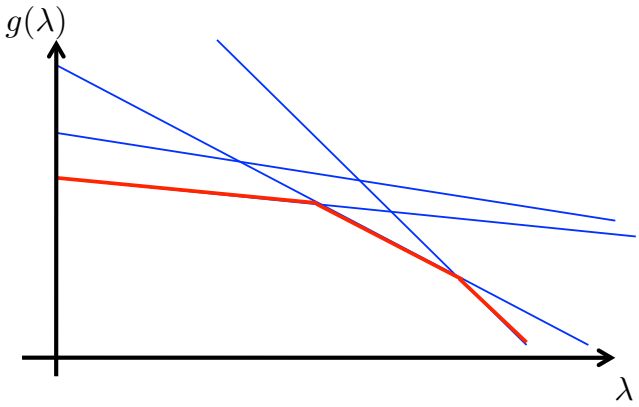

Figure 2.1: Example: Lagrangian with one inequality constraint, $L(x, \lambda) = f_0(x) + \lambda f_1(x)$, where x here can take one of four values for ease of illustration. The infimum of the resulting set of four affine functions is concave in λ.

Proposition 3. *When $\lambda \succeq 0$, then for all ν we have*

$$g(\lambda, \nu) \leq p^*. \tag{2.2}$$

Proof. Assume $\tilde{x} \in \mathcal{D}$ is feasible, i.e. $f_i(\tilde{x}) \leq 0$, $h_j(\tilde{x}) = 0$, and assume $\lambda \succeq 0$. Then

$$\sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{j=1}^{r} \nu_j h_j(\tilde{x}) \leq 0$$

and so

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

This holds for every feasible $\tilde{x}$, and thus also for x^*, hence (2.2) holds. □

Lagrangian can be interpreted as a lower bound on the original optimization problem. Idealy we would write the problem (2.1) as the unconstrained problem

$$\text{minimize } f_0(x) + \sum_{i=1}^{m} I_-\left(f_i(x)\right) + \sum_{j=1}^{r} I_0\left(h_j(x)\right),$$

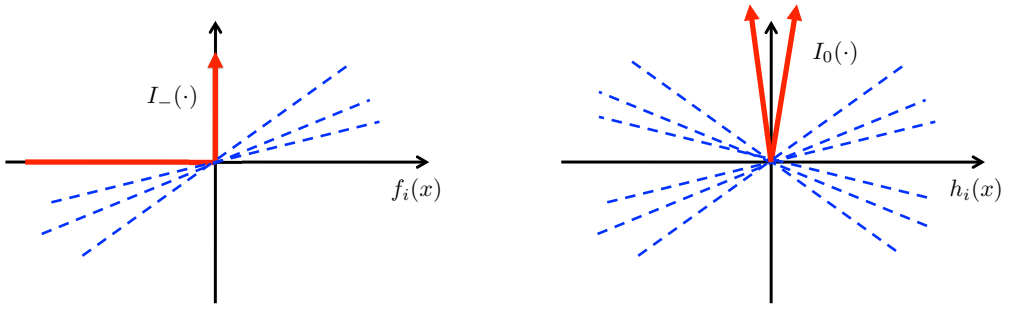

Figure 2.2: Linear lower bounds on indicator functions. Blue functions represent linear lower bounds for different slopes λ and ν, for the inequality and equality constraints, respectively.

where

$$I_-(u) = \begin{cases} 0 & u \leq 0 \\ \infty & u > 0, \end{cases} \qquad I_0(u) = \begin{cases} 0 & u = 0 \\ \infty & u \neq 0, \end{cases}$$

i.e. giving an infinite penalty when any constraint is violated. Instead of these infinite penalty constraints (which are hard to optimize), we replace the constraints with a set of soft linear constraints, as shown in Fig. 2.2. It is now clear why λ must be positive for the inequality constraint: a negative λ would not yield a lower bound. Note also that as well as being penalized for $f_i > 0$, the linear lower bounds reward us for achieving $f_i < 0$. This is illustrated in Fig. 2.3.

2.1.2 The dual problem

The dual problem attempts to find the best lower bound $g(\lambda, \nu)$ on the optimal solution p^* of (2.1). This results in the **Lagrange dual problem**

$$\begin{array}{ll} \text{maximize} & g(\lambda, \nu) \\ \text{subject to} & \lambda \succeq 0. \end{array} \tag{2.3}$$

Denote by (λ^*, ν^*) the arguments optimizing (2.3) and by d^* the optimal value of the dual problem. Note that (2.3) is always a convex optimization problem, since the function being maximized is concave and the constraint set is convex. The property of **weak duality** is immediate:

$$d^* \leq p^*.$$

The difference $p^* - d^*$ is called the **optimal duality gap**. If the duality gap is zero, then **strong duality** holds:

$$d^* = p^*.$$

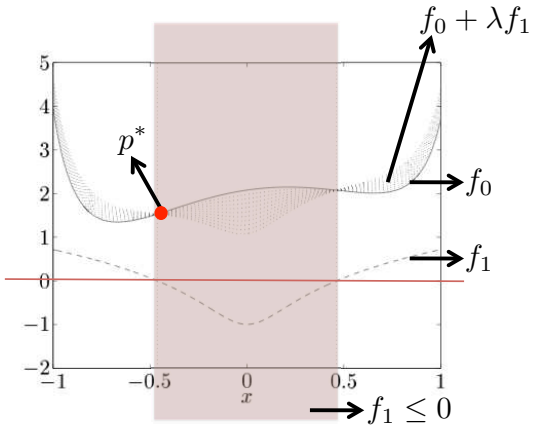

Figure 2.3: Illustration of the Lagrangian on a simple problem with one inequality constraint (from [6, Fig. 5.1]).

Conditions under which strong duality holds are called **constraint qualifications**. As an important case: strong duality holds if the primal problem is convex,[2] i.e. of the form

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \leq 0 && i = 1, \ldots, n \\ &&& Ax = b \end{aligned} \tag{2.4}$$

for convex $f_0, \ldots, f_m$, *and* if **Slater's condition** holds: there exists some *strictly* feasible point[3] $\tilde{x} \in \text{relint}(\mathcal{D})$ such that

$$f_i(\tilde{x}) < 0 \quad i = 1, \ldots, m \quad A\tilde{x} = b.$$

A weaker version of Slater's condition is sufficient for strong convexity when some of the constraint functions $f_1, \ldots, f_k$ are affine (note the inequality constraints are no longer strict):

$$f_i(\tilde{x}) \leq 0 \quad i = 1, \ldots, k \quad f_i(\tilde{x}) < 0 \quad i = k+1, \ldots, m \quad A\tilde{x} = b.$$

A proof of this result is given in [6, Section 5.3.2].

[2] Strong duality can also hold for non-convex problems: see e.g. [6, p. 229].

[3] We denote by $\text{relint}(\mathcal{D})$ the relative interior of the set $\mathcal{D}$. This looks like the interior of the set, but is non-empty even when the set is a subspace of a larger space. See [6, Section 2.1.3] for the formal defintion.

2.1.3 A saddlepoint/game characterization of weak and strong duality

In this section, we ignore equality constraints for ease of discussion. We write the solution to the primal problem as an optimization

$$\begin{aligned}\sup_{\lambda \succeq 0} L(x,\lambda) &= \sup_{\lambda \succeq 0}\left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x)\right) \\ &= \begin{cases} f_0(x) & f_i(x) \leq 0,\ i = 1,\ldots,m \\ \infty & \text{otherwise.} \end{cases}\end{aligned}$$

In other words, we recover the primal problem when the inequality constraint holds, and get infinity otherwise. We can therefore write

$$p^* = \inf_x \sup_{\lambda \succeq 0} L(x,\lambda).$$

We already know

$$d^* = \sup_{\lambda \succeq 0} \inf_x L(x,\lambda).$$

Weak duality therefore corresponds to the **max-min inequality**:

$$\sup_{\lambda \succeq 0} \inf_x L(x,\lambda) \leq \inf_x \sup_{\lambda \succeq 0} L(x,\lambda). \tag{2.5}$$

which holds for general functions, and not just $L(x,\lambda)$. Strong duality occurs at a saddlepoint, and the inequality becomes an equality.

There is also a game interpretation: $L(x,\lambda)$ is a sum that must be paid by the person adjusting x to the person adjusting λ. On the right hand side of (2.5), player x plays first. Knowing that player 2 (λ) will maximize their return, player 1 (x) chooses their setting to give player 2 the worst possible options over all λ. The max-min inequality says that whoever plays second has the advantage.

2.1.4 Optimality conditions

If the primal is equal to the dual, we can make some interesting observations about the duality constraints. Denote by x^* the optimum solution of the original problem (the minimum of f_0 under its constraints), and by (λ^*, ν^*) the solutions to the dual. Then

$$\begin{aligned} f_0(x^*) &= g(\lambda^*, \nu^*) \\ &\underset{\text{(a)}}{=} \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i^* f_i(x) + \sum_{i=1}^{r} \nu_i^* h_i(x)\right) \\ &\underset{\text{(b)}}{\leq} f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* f_i(x^*) + \sum_{i=1}^{r} \nu_i^* h_i(x^*) \\ &\leq f_0(x^*), \end{aligned}$$

where in (a) we use the definition of g, in (b) we use that $\inf_{x\in\mathcal{D}}$ of the expression in the parentheses is necessarily no greater than its value at x^*, and the last line we use that at (x^*, λ^*, ν^*) we have $\lambda^* \succeq 0$, $f_i(x^*) \leq 0$, and $h_i(x^*) = 0$. From this chain of reasoning, it follows that

$$\sum_{i=1}^{m} \lambda_i^* f_i(x^*) = 0, \tag{2.6}$$

which is the condition of **complementary slackness**. This means

$$\begin{aligned} \lambda_i^* > 0 \quad &\implies \quad f_i(x^*) = 0, \\ f_i(x^*) < 0 \quad &\implies \quad \lambda_i^* = 0. \end{aligned}$$

Consider now the case where the functions f_i, h_i are differentiable, and the duality gap is zero. Since x^* minimizes $L(x, \lambda^*, \nu^*)$, the derivative at x^* should be zero,

$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{r} \nu_i^* \nabla h_i(x^*) = 0.$$

We now gather the various conditions for optimality we have discussed. The **KKT conditions** for the primal and dual variables (x, λ, ν) are

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

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.

2.2 Support vector classification

2.2.1 The linearly separable case

We first consider problem of classifying two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error. This is illustrated in Fig. (2.4) for a 2-dimensional classification problem. As can be seen, there are infinitely many possible hyperplanes that solve this problem: the question is then: which one to choose? The principle behind support vector machines is that we choose the one which has the largest margin: i.e. the *smallest* distance from each class to the separating hyperplane is maximized.

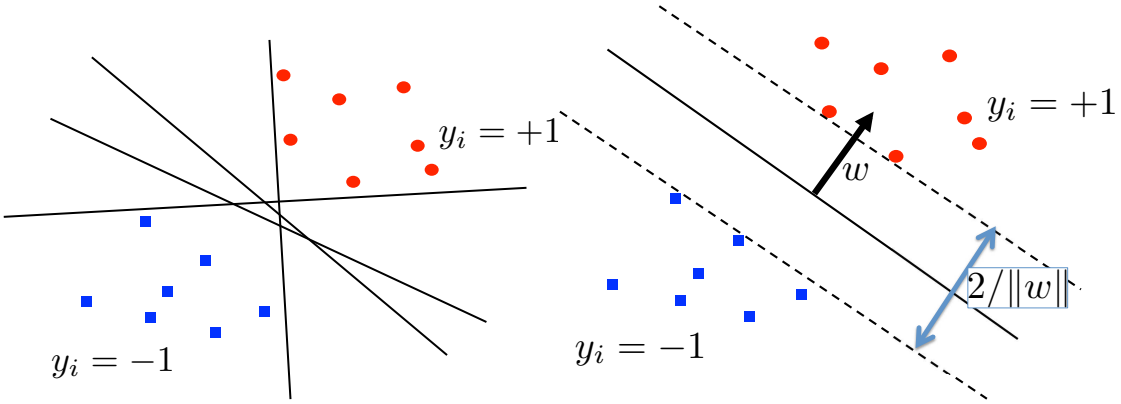

Figure 2.4: The linearly separable case. There are many linear separating hyperplanes, but only one maximum margin separating hyperplane.

This problem can be expressed as follows:[4]

$$\max_{w,b} (\text{margin}) = \max_{w,b} \left(\frac{2}{\|w\|} \right) \tag{2.8}$$

subject to

$$\begin{cases} \min \left(w^\top x_i + b \right) = 1 & i \ : \ y_i = +1, \\ \max \left(w^\top x_i + b \right) = -1 & i \ : \ y_i = -1. \end{cases} \tag{2.9}$$

The resulting classifier is

$$y = \text{sign}(w^\top x + b),$$

where sign takes value $+1$ for a positive argument, and -1 for a negative argument (its value at zero is not important, since for non-pathological cases we will not need to evaluate it there). We can rewrite to obtain

$$\max_{w,b} \frac{1}{\|w\|} \quad \text{or} \quad \min_{w,b} \|w\|^2$$

subject to

$$y_i(w^\top x_i + b) \geq 1. \tag{2.10}$$

[4]It's easy to see why the equation below is the margin (the distance between the positive and negative classes): consider two points exactly opposite each other and located on the margins, such that $(x_i - x_j) = \beta w$ for some scalar β (where we recall w is orthogonal to the decision boundary, hence aligned with $x_i - x_j$). Then the distance between them (which is the width of the margin) is

$$\|x_i - x_j\| = (x_i - x_j)^\top \frac{(x_i - x_j)}{\|x_i - x_j\|} = (x_i - x_j)^\top \frac{w}{\|w\|} \tag{2.7}$$

Subtracting the two equations in the constraints (2.9) from each other, we get

$$w^\top (x_i - x_j) = 2.$$

Substituting this into (2.7) proves the result.

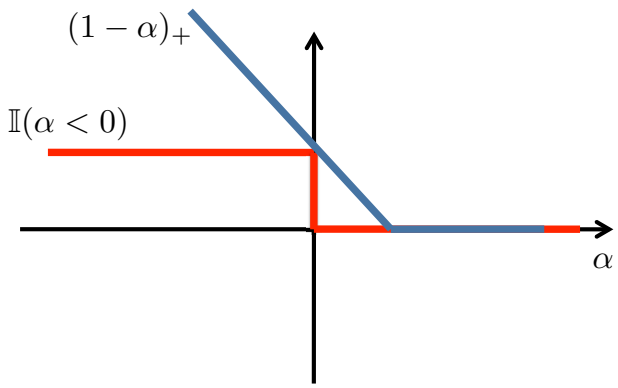

Figure 2.5: The hinge loss is an upper bound on the 0/1 loss.

2.2.2 When no linear separator exists (or we want a larger margin)

If the classes are not linearly separable, we may wish to allow a certain number of errors in the classifier (points within the margin, or even on the wrong side of the decision boudary). We therefore want to trade off such "margin errors" vs maximising the margin. Ideally, we would optimise

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

where C controls the tradeoff between maximum margin and loss (the factor of $1/2$ is to simplify the algebra later, and is not important: we can adjust C accordingly). This is a combinatorial optimization problem, which would be very expensive to solve. Instead, we replace the indicator function with a convex upper bound,

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

We use the hinge loss,

$$h(\alpha) = (1-\alpha)_+ = \begin{cases} 1-\alpha & 1-\alpha > 0 \\ 0 & \text{otherwise.} \end{cases}$$

although obviously other choices are possible (e.g. a quadratic upper bound). See Figure 2.5.

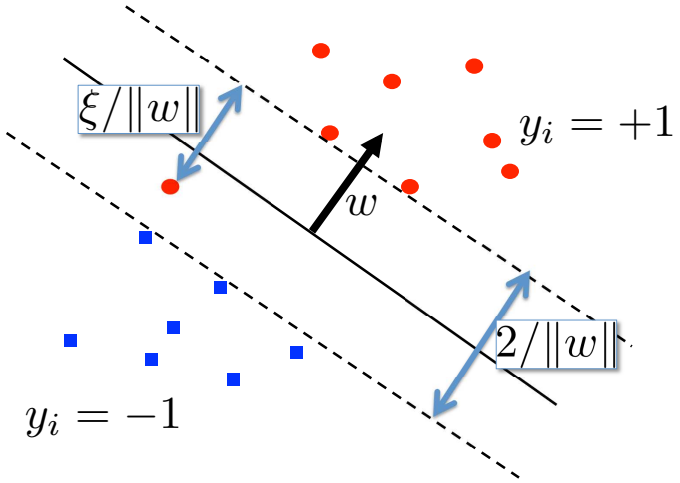

Figure 2.6: The nonseparable case. Note the red point which is a distance $\xi/\|w\|$ from the margin.

Substituting in the hinge loss, we get

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

or equivalently the constrained problem

$$\min_{w,b,\xi} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \right) \tag{2.11}$$

subject to[5]

$$\xi_i \geq 0 \qquad y_i \left(w^\top x_i + b\right) \geq 1 - \xi_i$$

(compare with (2.10)). See Figure 2.6.

Now let's write the Lagrangian for this problem, and solve it.

$$L(w, b, \xi, \alpha, \lambda) = \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(w^\top x_i + b\right) - \xi_i\right) + \sum_{i=1}^{n} \lambda_i (-\xi_i) \tag{2.12}$$

with dual variable constraints

$$\alpha_i \geq 0, \qquad \lambda_i \geq 0.$$

We minimize wrt the primal variables w, b, and ξ.

[5] To see this, we can write it as $\xi_i \geq 1 - y_i \left(w^\top x_i + b\right)$. Thus either $\xi_i = 0$, and $y_i \left(w^\top x_i + b\right) \geq 1$ as before, or $\xi_i > 0$, in which case to minimize (2.11), we'd use the smallest possible ξ_i satisfying the inequality, and we'd have $\xi_i = 1 - y_i \left(w^\top x_i + b\right)$.

Derivative wrt w:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \qquad w = \sum_{i=1}^{n} \alpha_i y_i x_i. \tag{2.13}$$

Derivative wrt b:

$$\frac{\partial L}{\partial b} = \sum_{i} y_i \alpha_i = 0. \tag{2.14}$$

Derivative wrt ξ_i:

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i = 0 \qquad \alpha_i = C - \lambda_i. \tag{2.15}$$

We can replace the final constraint by noting $\lambda_i \geq 0$, hence

$$\alpha_i \leq C.$$

Before writing the dual, we look at what these conditions imply about the scalars α_i that define the solution (2.13) due to complementary slackness.

Non-margin SVs: $\alpha_i = C > 0$:

1. We immediately have $1 - \xi_i = y_i \left(w^\top x_i + b\right)$.
2. Also, from condition $\alpha_i = C - \lambda_i$, we have $\lambda_i = 0$, hence $\xi_i \geq 0$.

Margin SVs: $0 < \alpha_i < C$:

1. We again have $1 - \xi_i = y_i \left(w^\top x_i + b\right)$
2. This time, from $\alpha_i = C - \lambda_i$, we have $\lambda_i > 0$, hence $\xi_i = 0$.

Non-SVs: $\alpha_i = 0$

1. This time we have: $y_i \left(w^\top x_i + b\right) \geq 1 - \xi_i$
2. From $\alpha_i = C - \lambda_i$, we have $\lambda_i > 0$, hence $\xi_i = 0$.

This means that the solution is *sparse*: all the points which are not either on the margin, or "margin errors", contribute nothing to the solution. In other words, only those points on the decision boundary, or which are margin errors, contribute. Furthermore, the influence of the non-margin SVs is bounded, since their weight cannot exceed C: thus, severe outliers will not overwhelm the solution.

We now write the dual function, by substituting equations (2.13), (2.14), and (2.15) into (2.12), to get

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

Thus, our goal is to maximize the dual,

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

subject to the constraints

$$
0 \le \alpha_i \le C, \quad \sum_{i=1}^{n} y_i\alpha_i = 0.
$$

So far we have defined the solution for w, but not for the offset b. This is simple to compute: for the margin SVs, i.e., those x_i for which $0 < \alpha_i < C$, we have $1 = y_i\left(w^\top x_i + b\right)$. Thus, we can obtain b from any of these, or take an average for greater numerical stability.

2.2.3 The ν-SVM

It can be hard to interpret C. Therefore we modify the formulation to get a more intuitive parameter. Again, we drop b for simplicity. Solve

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

subject to

$$
\begin{aligned}
\rho &\ge 0 \\
\xi_i &\ge 0 \\
y_i w^\top x_i &\ge \rho - \xi_i,
\end{aligned}
$$

where we see that we now optimize the margin width ρ. Thus, rather than choosing C, we now choose ν as a hyperparameter; the meaning of the latter will become clear shortly.

The Lagrangian is

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

for $\alpha_i \geq 0$, $\beta_i \geq 0$, and $\gamma \geq 0$. Differentiating wrt each of the primal variables w, ξ, ρ, and setting to zero, we get

$$\begin{aligned} w &= \sum_{i=1}^{n}\alpha_i y_i x_i \\ \alpha_i + \beta_i &= \frac{1}{n} \qquad (2.16) \\ \nu &= \sum_{i=1}^{n}\alpha_i - \gamma \qquad (2.17) \end{aligned}$$

From $\beta_i \geq 0$, equation (2.16) implies

$$0 \leq \alpha_i \leq n^{-1}.$$

From $\gamma \geq 0$ and (2.17), we get

$$\nu \leq \sum_{i=1}^{n}\alpha_i.$$

We typically have $\rho > 0$ at the global solution (i.e. non-zero margin) and hence $\gamma = 0$, and (2.17) becomes

$$\sum_{i=1}^{n}\alpha_i = \nu. \qquad (2.18)$$

Complementary slackness conditions now lead to a very convenient interpretation of parameter ν. In particular, if we denote by $N(\alpha)$ the set of non-margin support vectors, i.e. margin errors, and by $M(\alpha)$ the set of margin support vectors, then (exercise):

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

Thus 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 - tuning ν is hence much more interpretable than tuning C.

Substituting into the Lagrangian, we can also obtain the dual formulation of ν-SVM, i.e.

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

Thus, we must maximize

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

subject to

$$\sum_{i=1}^{n}\alpha_i \geq \nu \qquad 0 \leq \alpha_i \leq \frac{1}{n}.$$

3 Kernel Methods

3.1 Feature Maps and Feature Spaces

Kernel methods are a versatile algorithmic framework which allows construction of nonlinear machine learning algorithms (for a variety of both supervised and unsupervised learning tasks: clustering, dimensionality reduction, classification, regression) by employing linear tools in a nonlinearly transformed feature space. Let us first recall the definition of an abstract inner product, which is central to kernel methods.

Definition 4. [Inner product] Let $\mathcal{H}$ be a vector space over $\mathbb{R}$. A function $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is said to be *an inner product* on $\mathcal{H}$ if

1. $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle_{\mathcal{H}} = \alpha_1 \langle f_1, g \rangle_{\mathcal{H}} + \alpha_2 \langle f_2, g \rangle_{\mathcal{H}}$
2. $\langle f, g \rangle_{\mathcal{H}} = \langle g, f \rangle_{\mathcal{H}}$
3. $\langle f, f \rangle_{\mathcal{H}} \geq 0$ and $\langle f, f \rangle_{\mathcal{H}} = 0$ if and only if $f = 0$.

We can define a norm using the inner product as $\|f\|_{\mathcal{H}} := \sqrt{\langle f, f \rangle_{\mathcal{H}}}$. A *Hilbert space* is a vector space on which an inner product is defined, along with an additional technical condition.[1] We are now ready to define the notion of a *kernel*.

Definition 5. Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *kernel* if there exists a Hilbert space and a map $\varphi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$,

$$k(x, x') := \left\langle \varphi(x), \varphi(x') \right\rangle_{\mathcal{H}}.$$

We will call such $\mathcal{H}$ of kernel k a *feature space* and the map φ will be called a *feature map*. Note that we imposed almost no conditions on $\mathcal{X}$: in particular, we do not require there to be an inner product defined on the elements of $\mathcal{X}$. The case of text documents is an instructive example: one cannot take an inner product between two books, but can take an inner product between features of the text in those books.

Clearly, a single kernel can correspond to multiple pairs of underlying feature maps and feature spaces. For a simple example, consider $\mathcal{X} := \mathbb{R}^p$:

$$\phi_1(x) = x \qquad \text{and} \qquad \phi_2(x) = \left[\frac{x_1}{\sqrt{2}}, \cdots, \frac{x_p}{\sqrt{2}}, \frac{x_1}{\sqrt{2}}, \cdots, \frac{x_p}{\sqrt{2}} \right]^\top.$$

[1] Specifically, a Hilbert space must be *complete*, i.e. it must contain the limits of all Cauchy sequences with respect to the norm defined by its inner product.

Both ϕ_1 and ϕ_2 are valid feature maps (with feature spaces $\mathcal{H}_1 = \mathbb{R}^p$ and $\mathcal{H}_2 = \mathbb{R}^{2p}$) of kernel $k(x, x') = x^\top x'$.

It turns out that all kernel functions (defined as inner products between some features) are *positive definite*.

Definition 6. [Positive definite functions] A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive definite if $\forall n \geq 1$, $\forall (a_1, \dots a_n) \in \mathbb{R}^n$, $\forall (x_1, \dots, x_n) \in \mathcal{X}^n$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(x_i, x_j) \geq 0.$$

The function $k(\cdot, \cdot)$ is *strictly* positive definite if for mutually distinct x_i, the equality holds only when all the a_i are zero.[2]

Every inner product is a positive definite function, and so is every inner product between feature maps.

Lemma 7. *Let $\mathcal{H}$ be any Hilbert space, $\mathcal{X}$ a non-empty set and $\phi : \mathcal{X} \to \mathcal{H}$. Then $k(x, y) := \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ is a positive definite function.*

Proof.

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(x_i, x_j) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle a_i \phi(x_i), a_j \phi(x_j) \rangle_{\mathcal{H}} \\ &= \left\| \sum_{i=1}^{n} a_i \phi(x_i) \right\|_{\mathcal{H}}^2 \geq 0. \end{aligned}$$

□

3.2 Reproducing Kernel Hilbert Spaces

We have introduced the notation of feature spaces, and kernels on these feature spaces. What's more, we've determined that these kernels are positive definite. In this section, we use these kernels to define *functions* on $\mathcal{X}$. The space of such functions is known as a reproducing kernel Hilbert space (RKHS).

Definition 8. [Reproducing kernel] Let $\mathcal{H}$ be a *Hilbert space of functions* $f \colon \mathcal{X} \to \mathbb{R}$ defined on a non-empty set $\mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called *a reproducing kernel* of $\mathcal{H}$ if it satisfies

- $\forall x \in \mathcal{X}$, $k_x = k(\cdot, x) \in \mathcal{H}$,
- $\forall x \in \mathcal{X}$, $\forall f \in \mathcal{H}$, $\langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x)$ (*the reproducing property*).

[2] The corresponding terminology used for matrices is "positive semi-definite" vs "positive definite".

If $\mathcal{H}$ has a reproducing kernel, it is called a reproducing kernel Hilbert space (RKHS).

In particular, note that for any $x, y \in \mathcal{X}$, reproducing kernel satisfies $k(x,y) = \langle k(\cdot,y), k(\cdot,x)\rangle_{\mathcal{H}} = \langle k(\cdot,x), k(\cdot,y)\rangle_{\mathcal{H}}$. Thus, reproducing kernel is clearly a kernel, i.e. an inner product between features with a feature space $\mathcal{H}$ and a feature map $\phi\colon x \mapsto k(\cdot,x)$. This way of writing feature mapping is called the *canonical feature map.* Note that these features are not specified explicitly in a vector form, but rather as functions on $\mathcal{X}$.

We have seen that any reproducing kernel is a kernel and that every kernel is a positive definite function. Remarkably, *Moore-Aronszajn theorem* [4] shows that *for every positive definite function k, there exists a unique RKHS with kernel k.* The theorem is outside of the scope of this course, but it provides an insight into the structure of the RKHS corresponding to k. It turns out RKHS can be written as $\overline{\operatorname{span}\{k(\cdot,x) : x \in \mathcal{X}\}}$, i.e. the space of all linear combinations of canonical features, completed with respect to an inner product on these linear combinations defined as

$$\left\langle \sum_{i=1}^{r} \alpha_i k(\cdot,x_i), \sum_{j=1}^{s} \beta_j k(\cdot,y_j) \right\rangle := \sum_{i=1}^{r}\sum_{j=1}^{s} \alpha_i\beta_j k(x_i,y_j).$$

Thus, all three notions: (1) reproducing kernel, (2) kernel as inner product between features and (3) positive definite function, are equivalent. Recall that the feature space of a kernel is not unique - but its RKHS (feature space as a space of functions) is - we will henceforth denote the RKHS of kernel k by $\mathcal{H}_k$. For example, for the *linear kernel* $k(x,y) = x^\top y$ considered earlier, many possible feature representations exist but the canonical feature representation that associates to each x the function $k(\cdot,x)\colon y \mapsto x^\top y$ is what determines the structure of its RKHS. In particular, linear kernel $k(x,y) = x^\top y$ corresponds to the RKHS $\mathcal{H}_k$ which is the space of all linear functions $f(x) = w^\top x$ (*why*?).

3.3 Representer Theorem

Now that we have defined an RKHS, we can consider it as a hypothesis class for empirical risk minimisation (ERM). In particular, we are looking for the function f^* in the RKHS $\mathcal{H}_k$ which solves the regularised ERM problem

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

for empirical risk $\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i), x_i)$, a loss function $L\colon \mathcal{Y}\times\mathcal{Y}\times\mathcal{X} \to \mathbb{R}_+$ and any non-decreasing function Ω.

Theorem 9. *There is a solution to*

$$\min_{f\in\mathcal{H}_k} \hat{R}(f) + \Omega\left(\|f\|^2_{\mathcal{H}_k}\right) \tag{3.1}$$

that takes the form $f^* = \sum_{i=1}^{n} \alpha_i k(\cdot,x_i)$. If Ω is strictly increasing, all solutions have this form.

Proof. Let f be any minimiser of (3.1). Denote by f_s the projection of f onto the subspace

$$\operatorname{span}\{k(\cdot, x_i) : i = 1, \ldots, n\}$$

such that

$$f = f_s + f_\perp,$$

where $f_s = \sum_{i=1}^n \alpha_i k(\cdot, x_i)$ and $f_\perp$ is orthogonal to the subspace $\operatorname{span}\{k(\cdot, x_i) : i = 1, \ldots, n\}$. Since

$$\|f\|_{\mathcal{H}_k}^2 = \|f_s\|_{\mathcal{H}_k}^2 + \|f_\perp\|_{\mathcal{H}_k}^2 \geq \|f_s\|_{\mathcal{H}_k}^2,$$

we have

$$\Omega\left(\|f\|_{\mathcal{H}_k}^2\right) \geq \Omega\left(\|f_s\|_{\mathcal{H}_k}^2\right).$$

On the other hand, the individual terms $f(x_i)$ in the loss are given by

$$f(x_i) = \langle f, k(\cdot, x_i)\rangle_{\mathcal{H}_k} = \langle f_s + f_\perp, k(\cdot, x_i)\rangle_{\mathcal{H}_k} = \langle f_s, k(\cdot, x_i)\rangle_{\mathcal{H}_k} = f_s(x_i),$$

so

$$L(y_i, f(x_i), x_i) = L(y_i, f_s(x_i), x_i) \quad \forall i = 1, \ldots, n.$$

and thus empirical risks must be the same: $\hat{R}(f) = \hat{R}(f_s)$. Thus f_s is also a minimiser of (3.1) and if Ω is strictly increasing, it must be that $f_\perp = 0$. □

We see that the key parts of the theorem are the fact that the empirical risk only depends on the components of f lying in the subspace spanned by the canonical features and that the regulariser $\Omega(\cdot)$ is minimised when $f = f_s$ (adding additional orthogonal components to the function makes it more complex but does not change the empirical risk). Moreover, if Ω is strictly increasing, then $\|f_\perp\|_{\mathcal{H}_k} = 0$ is required at the minimum.

3.4 Operations with Kernels

Kernels can be combined and modified to get new kernels. For example,

Lemma 10. *[Sums of kernels are kernels] Given $\alpha > 0$ and k, k_1 and k_2 all kernels on $\mathcal{X}$, then αk and $k_1 + k_2$ are kernels on $\mathcal{X}$.*

To prove the above, just check *positive definiteness*. Note that a difference between two kernels need not be a kernel: if $k_1(x, x) - k_2(x, x) < 0$, then condition 3 of inner product definition 4 may be violated.

Lemma 11. *[Mappings between spaces] Let $\mathcal{X}$ and $\widetilde{\mathcal{X}}$ be non-empty sets, and define a map $A : \mathcal{X} \to \widetilde{\mathcal{X}}$. Define the kernel k on $\widetilde{\mathcal{X}}$. Then $k(A(x), A(x'))$ is a kernel on $\mathcal{X}$.*

Lemma 12. *[Products of kernels are kernels] Given k on $\mathcal{X}$ and l on $\mathcal{Y}$, then*

$$\kappa\left((x, y), (x', y')\right) = k\left(x, x'\right) l\left(y, y'\right)$$

is a kernel on $\mathcal{X} \times \mathcal{Y}$. Moreover, if $\mathcal{X} = \mathcal{Y}$, then

$$\kappa\left(x, x'\right) = k\left(x, x'\right) l\left(x, x'\right)$$

is a kernel on $\mathcal{X}$.

The general proof would require some technical details about Hilbert space tensor products, but the main idea can be understood with some simple linear algebra. We consider the case where $\mathcal{H}$ corresponding to k is $\mathbb{R}^M$, and $\mathcal{G}$ corresponding to l is $\mathbb{R}^N$. Write $k\left(x, x'\right) = \varphi(x)^\top \varphi(x')$ and $l\left(y, y'\right) = \psi(y)^\top \psi(y')$. We will use that a notion of inner product between matrices $A \in \mathbb{R}^{M \times N}$ and $B \in \mathbb{R}^{M \times N}$ is given by

$$\langle A, B \rangle = \operatorname{trace}(A^\top B). \tag{3.2}$$

Then

$$\begin{aligned} k\left(x, x'\right) l\left(y, y'\right) &= \varphi(x)^\top \varphi(x') \psi(y')^\top \psi(y) \\ &= \operatorname{tr}(\psi(y) \varphi(x)^\top \varphi(x') \psi(y')^\top) \\ &= \left\langle \varphi(x) \psi(y)^\top, \varphi(x') \psi(y')^\top \right\rangle, \end{aligned}$$

thus we can define features $A(x, y) = \varphi(x)\psi(y)^\top$ of the product kernel.

The sum and product rules allow us to define a huge variety of kernels.

Lemma 13. *[Polynomial kernels] Let $x, x' \in \mathbb{R}^p$ for $p \geq 1$, and let $m \geq 1$ be an integer and $c \geq 0$. Then*

$$k(x, x') := \left(\langle x, x' \rangle + c\right)^m$$

is a valid kernel.

To prove: expand out this expression into a sum (with non-negative scalars) of kernels $\langle x, x' \rangle$ raised to integer powers. These individual terms are valid kernels by the product rule.

Can we extend this combination of sum and product rule to sums with infinitely many terms? Consider for example the exponential function applied to an inner product $k(x, x') = \exp\left(\langle x, x' \rangle\right)$. Since addition and multiplication preserve positive definiteness and since all the coefficients in the Taylor series expansion of the exponential function are nonnegative, $k_m(x, x') = \sum_{r=1}^{m} \frac{\langle x, x' \rangle^r}{r!}$ is a valid kernel $\forall m \in \mathbb{N}$. Fix some $\{\alpha_i\}$ and $\{x_i\}$. Then $A_m = \sum_{i,j} \alpha_i \alpha_j k_m(x_i, x_j) \geq 0$ $\forall m$ since k_m is positive definite. But $A_m \to \sum_{i,j} \alpha_i \alpha_j \exp\left(\langle x_i, x_j \rangle\right)$ as $m \to \infty$, so $\sum_{i,j} \alpha_i \alpha_j \exp\left(\langle x_i, x_j \rangle\right) \geq 0$ as well. Thus, $\exp\left(\langle x, x' \rangle\right)$ is also a valid kernel (it is called *exponential kernel*). We may combine all the results above (*exercise*) to show that the following kernel, in practice widely used and known under various names: *Gaussian, Gaussian RBF, squared exponential* or *exponentiated quadratic,* is valid on $\mathbb{R}^p$:

$$k(x, x') := \exp\left(-\frac{1}{2\gamma^2} \left\|x - x'\right\|^2\right).$$

The RKHS of this kernel is infinite-dimensional. Moreover, if the domain $\mathcal{X}$ is a compact subset of $\mathbb{R}^p$, its RKHS is dense in the space of all bounded continuous functions with

respect to the uniform norm. Despite that, since all functions in its RKHS are infinitely differentiable, Gaussian kernel is often considered to be excessively smooth. A less smooth alternative is the Matérn family, given by

$$k(x,x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}}{\gamma}\left\|x-x'\right\|\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{\gamma}\left\|x-x'\right\|\right), \quad \nu>0, \gamma>0,$$

where K_ν is the modified Bessel function of the second kind of order ν. The Matérn kernels corresponding to the values $\nu = s + \frac{1}{2}$ for non-negative integers s take a simpler form, in particular:

- $\nu = 1/2$: $k(x,x') = \exp\left(-\frac{1}{\gamma}\left\|x-x'\right\|\right)$,
- $\nu = 3/2$: $k(x,x') = \left(1+\frac{\sqrt{3}}{\gamma}\left\|x-x'\right\|\right)\exp\left(-\frac{\sqrt{3}}{\gamma}\left\|x-x'\right\|\right)$,
- $\nu = 5/2$: $k(x,x') = \left(1+\frac{\sqrt{5}}{\gamma}\left\|x-x'\right\|+\frac{5}{3\gamma^2}\left\|x-x'\right\|^2\right)\exp\left(-\frac{\sqrt{5}}{\gamma}\left\|x-x'\right\|\right)$.

For $\nu = s + \frac{1}{2}$, its RKHS consists of $s+1$ times differentiable functions with square integrable derivatives of order up to $s+1$. Moreover, the RKHS norms directly penalize the derivatives of f, e.g. for $\nu = 3/2$ and in one dimension, it can be shown that

$$\|f\|^2_{\mathcal{H}_k} \propto \int f''(x)^2 dx + \frac{6}{\gamma^2}\int f'(x)^2 dx + \frac{9}{\gamma^4}\int f(x)^2 dx.$$

As $\nu \to \infty$, Matérn kernel converges to the Gaussian RBF, i.e. $k(x,x') = \exp\left(-\frac{1}{2\gamma^2}\left\|x-x'\right\|^2\right)$.

Another popular choice of a kernel is *rational quadratic* which arises as a scale mixture of Gaussian kernels. In particular, consider Gaussian RBF parametrisation $k_\theta(x,x') = \exp\left(-\theta\left\|x-x'\right\|^2\right)$ and a Gamma density placed on θ, i.e. $p(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{\alpha-1}\exp(-\beta\theta)$, with shape α and rate β. Then, we define

$$\begin{aligned}
\kappa(x,x') &= \int_0^\infty k_\theta(x,x')p(\theta)d\theta \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)}\int_0^\infty \exp\left(-\theta\left(\left\|x-x'\right\|^2+\beta\right)\right)\theta^{\alpha-1}d\theta \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)}\frac{\Gamma(\alpha)}{\left(\left\|x-x'\right\|^2+\beta\right)^\alpha} \\
&= \left(1+\frac{\left\|x-x'\right\|^2}{\beta}\right)^{-\alpha}.
\end{aligned}$$

Rational quadratic RKHS models functions which vary smoothly across multiple length-scales. If we write $\beta = 2\alpha\gamma^2$ and let $\alpha \to \infty$ we again recover Gaussian RBF, i.e. $\exp\left(-\frac{1}{2\gamma^2}\left\|x-x'\right\|^2\right)$.

3.5 Kernel SVM

We can straightforwardly define a maximum margin classifier, i.e. a Support Vector Machine (SVM) in the RKHS. We write the original hinge loss formulation of the regularized empirical risk minimization (ignoring the offset b here for simplicity[3]):

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

for the RKHS $\mathcal{H}$ with kernel $k(x, x')$. This "kernelized" SVM satisfies the assumption of the representer theorem, so we are looking for the solutions of the form

$$w = \sum_{i=1}^{n} \beta_i k(x_i, \cdot). \tag{3.3}$$

In this case, maximizing the margin in the RKHS is equivalent to minimizing $\|w\|_{\mathcal{H}}^2$: as we have seen, for many RKHSs (e.g. the RKHS corresponding to a Gaussian kernel), this corresponds to enforcing smoothness of the learned functions.

Substituting (3.3) and introducing the ξ_i variables as before, we get

$$\min_{\beta, \xi} \left(\frac{1}{2}\beta^\top K \beta + C \sum_{i=1}^{n} \xi_i \right) \tag{3.4}$$

$$\text{subject to } \xi_i \geq 0 \qquad y_i \sum_{j=1}^{n} \beta_j k(x_i, x_j) \geq 1 - \xi_i$$

where the matrix K has i, jth entry $K_{ij} = k(x_i, x_j)$. Thus, the primal variables w are replaced with coefficients β. Note that the problem remains convex since matrix K is positive definite. With an easy calculation (left for exercise), we can verify that the dual takes the form

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

subject to the constraints

$$0 \leq \alpha_i \leq C,$$

and the decision function takes the form

$$w = \sum_{i=1}^{n} y_i \alpha_i k(x, \cdot).$$

This is analogous to the original dual SVM, with inner products replaced with the kernel k.

[3] Note that it suffices to add a constant feature or equivalently use the kernel $k(x, x') + 1$ to include the offset.

3.6 Kernel PCA

Kernel PCA is a popular nonlinear dimensionality reduction technique [24]. Assume we have a dataset $\{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$. Consider an explicit feature transformation $x \mapsto \varphi(x) \in \mathcal{H}$, and assume that we are interested in performing PCA in the feature space $\mathcal{H}$. Assume that the features $\{\varphi(x_i)\}_{i=1}^n$ are centred. Assume for the moment that the feature space is finite-dimensional, i.e. $\mathcal{H} = \mathbb{R}^M$. Then the $M \times M$ sample covariance matrix in the feature space is given by

$$\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^{n} \varphi(x_i)\varphi(x_i)^\top = \frac{1}{n-1} \Phi^\top \Phi,$$

where $\Phi \in \mathbb{R}^{n \times M}$ is the feature representation of the data. To perform PCA, recall that we are interested in solving the eigenvalue problem $\mathbf{S}v_m = \lambda_m v_m$, $m = 1, \ldots, M$, and we need the top $k \ll \min\{n, M\}$ eigenvectors v_m, $m = 1, \ldots, k$, to construct the PC projections $z_i^{(m)} = v_m^\top \varphi(x_i)$. A property analogous to the representer theorem holds here: whenever $\lambda_m > 0$, the eigenvectors lie in the linear span of feature vectors $\operatorname{span}\{\varphi(x_i) : i = 1, \ldots, n\}$, i.e.

$$v_m = \sum_{i=1}^{n} a_{mi} \varphi(x_i) \tag{3.5}$$

for some scalars a_{mi}. To see this, note that

$$\lambda_m v_m = \mathbf{S}v_m = \frac{1}{n-1} \sum_{i=1}^{n} \varphi(x_i) \left(\varphi(x_i)^\top v_m \right)$$

and since $\lambda_m > 0$, it suffices to take $a_{mi} = \frac{1}{\lambda_m (n-1)} \left(\varphi(x_i)^\top v_m \right)$ and clearly v_m has form (3.5). Thus eigenvectors can also be recovered in the *dual space*. Consider now the $n \times n$ kernel matrix $\mathbf{K}$ with $\mathbf{K}_{ij} = k(x_i, x_j) = \varphi(x_i)^\top \varphi(x_j)$. By substituting $v_m = \sum_{i=1}^n a_{mi} \varphi(x_i)$ back into the eigenvalue problem, we have:

$$\mathbf{S}v_m = \frac{1}{n-1} \sum_{i=1}^{n} \varphi(x_i) \sum_{\ell=1}^{n} a_{m\ell} k(x_i, x_\ell) = \lambda_m \sum_{i=1}^{n} a_{mi} \varphi(x_i).$$

To express the above in terms of the kernel matrix, we project both sides onto $\varphi(x_j)$, for each $j = 1, \ldots, n$. This gives

$$\frac{1}{n-1} \sum_{i=1}^{n} k(x_j, x_i) \sum_{\ell=1}^{n} a_{m\ell} k(x_i, x_\ell) = \lambda_m \sum_{i=1}^{n} a_{mi} k(x_j, x_i), \quad j = 1, \ldots, n,$$

which in matrix notation can be written as

$$\mathbf{K}^2 a_m = \lambda_m (n-1) \mathbf{K} a_m.$$

Assuming that $\mathbf{K}$ is invertible, a_m vectors can be found as the eigenvectors of the kernel matrix $\mathbf{K}$ with corresponding eigenvalues given by $\lambda_m(n-1)$.

But if we simply perform the eigendecomposition of $\mathbf{K}$, we will obtain n-dimensional eigenvectors of unit norm, and we are after the M-dimensional eigenvectors v_m of $\mathbf{S}$ which have unit norm. We see that $1 = v_m^\top v_m = a_m^\top \mathbf{K} a_m = \lambda_m(n-1) a_m^\top a_m$. Thus, if u_m denotes the m-th eigenvector of $\mathbf{K}$ with unit norm, to ensure that v_m has unit norm, we need to rescale $a_m = u_m/\sqrt{\lambda_m(n-1)}$. Now, we have an implicit representation of eigenvectors in terms of their dual coefficients. The PC projections are

$$z_i^{(m)} = v_m^\top \varphi(x_i) = \left(\sum_{j=1}^n a_{mj}\varphi(x_j)\right)^\top \varphi(x_i) = \sum_{j=1}^n a_{mj} k(x_j, x_i),$$

or equivalently, the m-th dimension of the PC projections is given by

$$\mathbf{z}^{(m)} \quad = \quad \mathbf{K} a_m = \lambda_m(n-1) a_m = \sqrt{\lambda_m(n-1)}\, u_m. \tag{3.6}$$

We have seen this before! Note that PC projections can be discovered from the SVD $\Phi = UDV^\top$ as either $\mathbf{Z} = \Phi V$ or $\mathbf{Z} = UD$. The latter expression is exactly (3.6), since u_m are the eigenvectors of kernel matrix $\mathbf{K}$ (i.e. the left singular vectors of the feature matrix Φ) and $D_{mm} = \sqrt{\lambda_m(n-1)}$ (*why*?). But note that the eigendecomposition of $\mathbf{K}$ and these projections do not require explicit feature transformations - thus, all the computation is happening in the dual representation and $\varphi(x_i)$ need not be computed, only the kernel matrix $\mathbf{K}$ with $\mathbf{K}_{ij} = k(x_i, x_j)$. The kernel formalism also allows us to compute the projection $v_m^\top \varphi(\tilde{x})$ of a new (previously unseen) data vector $\tilde{x} \in \mathbb{R}^p$ to the m-th kernel principal component using

$$\left(\sum_{i=1}^n a_{mi}\varphi(x_i)\right)^\top \varphi(\tilde{x}) = \sum_{i=1}^n a_{mi} k(x_i, \tilde{x}) = a_m^\top \mathbf{k}_{\tilde{x}},$$

where $\mathbf{k}_{\tilde{x}} = [k(x_1, \tilde{x}), \ldots, k(x_n, \tilde{x})]^\top$, so again no explicit feature transformations are needed.

Recall that the above all assumes that the features are centred, i.e. that $\frac{1}{n}\sum_{i=1}^n \varphi(x_i) = 0$, but if we are just given a kernel function $k(x, x')$, there is no reason to believe that the features would be centred. Fortunately, it is straightforward to transform *any* kernel matrix into a centred form. Note that the squared distance matrix in the feature space, i.e. matrix $\mathbf{D}$ for which

$$\mathbf{D}_{ij} = \|\varphi(x_i) - \varphi(x_j)\|_{\mathcal{H}}^2 = k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j)$$

can easily be recovered from the Gram/kernel matrix. In matrix form, In matrix form,

$$\mathbf{D} = \operatorname{diag}(\mathbf{K})\, \mathbf{1}^\top + \mathbf{1} \operatorname{diag}(\mathbf{K})^\top - 2\mathbf{K}.$$

But distances are invariant to centering and the Gram matrix corresponding to centred features can then also be recovered from the distance matrix (exercise).

3.7 Representation of probabilities in RKHS

We have seen that kernel methods effectively work on implicit representations of individual data points, via the canonical feature map $\phi\colon x \mapsto k(\cdot, x)$, such that every data point is represented as a point in the RKHS $\mathcal{H}_k$. One can similarly represent probability distributions P in the RKHSs by considering the *kernel mean embedding*

$$P \mapsto \mu_k(P) = \mathbb{E}_{X\sim P} k(\cdot, X) \in \mathcal{H}_k.$$

This is a potentially infinite-dimensional representation of P akin to a characteristic function of a probability distribution. Kernel mean embedding represents expectations over RKHS:

$$\langle f, \mu_k(P)\rangle_{\mathcal{H}_k} = \mathbb{E}_{X\sim P} f(X), \quad \forall f \in \mathcal{H}_k$$

and exists whenever $f \mapsto \mathbb{E}_{X\sim P} f(X)$ is a bounded functional. Note that this is always true if the kernel function itself is bounded, i.e. $k(x, y) \leq M < \infty \; \forall x, y$. Namely, by Cauchy-Schwarz

$$\mathbb{E}_{X\sim P} f(X) = \mathbb{E}_{X\sim P} \langle f, k(\cdot, X)\rangle \leq \|f\|_{\mathcal{H}_k} \mathbb{E}_{X\sim P} \|k(\cdot, X)\|_{\mathcal{H}_k} \leq \sqrt{M}\, \|f\|_{\mathcal{H}_k}$$

Such representation imposes a simple Hilbert space structure on probability distributions. In particular, inner products between kernel mean embeddings can be computed as

$$\langle \mu_k(P), \mu_k(Q)\rangle_{\mathcal{H}_k} = \mathbb{E}_{X\sim P}\mathbb{E}_{Y\sim Q} k(X, Y).$$

MMD. We can easily estimate the (squared) distances between probability measures induced by this RKHS representation since they correspond to simple expectations. Such distances are called *Maximum Mean Discrepancy (MMD):*

$$\begin{aligned} MMD_k^2(P, Q) &= \|\mu_k(P) - \mu_k(Q)\|_{\mathcal{H}_k}^2 \\ &= \mathbb{E}_{X,X'\overset{iid}{\sim}P} k(X, X') + \mathbb{E}_{Y,Y'\overset{iid}{\sim}Q} k(Y, Y') - 2\mathbb{E}_{X\sim P, Y\sim Q} k(X, Y), \end{aligned} \tag{3.7}$$

where X and X' denote independent copies of random variables with law P, and similarly for Y and Y'.

The name MMD comes from the following interpretation: it can also be written as the largest discrepancy between expectations of the unit norm RKHS functions with respect to two distributions (**exercise**):

$$\mathrm{MMD}_k(P, Q) = \sup_{f\in\mathcal{H}_k:\, \|f\|_{\mathcal{H}_k}\leq 1} \left|\mathbb{E}_{X\sim P} f(X) - \mathbb{E}_{Y\sim Q} f(Y)\right|.$$

As a consequence, the function f where the supremum is attained (which can be shown to be proportional to the difference between embeddings, i.e. $\mu_k(P) - \mu_k(Q)$, can be thought of as the *witness function* for the difference between distributions P and Q.

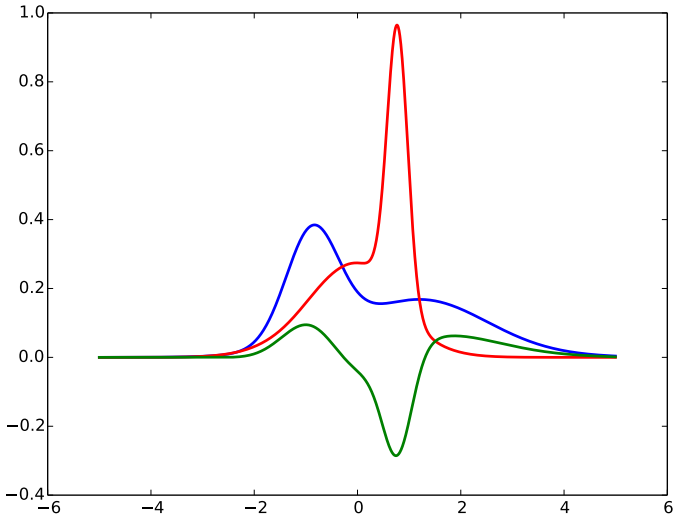

Figure 3.1: Witness function for a difference between two univariate densities

An example of such a witness function is shown in green in Fig. 3.1, where P and Q correspond to distributions on the real line whose densities are drawn in blue and red, respectively. We can see that the witness function is large in amplitude where the difference between two densities is large and it can thus be used to discover regions in the space where two distributions disagree.

For a large class of kernels, including Gaussian, Matern family and rational quadratic, MMD is a proper metric on probability distributions, in the sense that $MMD_k(P, Q) = 0$ implies $P = Q$. Such kernels are called *characteristic*. MMD is a popular probability metric, used for nonparametric hypothesis testing [13] and in various machine learning applications, e.g. training deep generative models [10]. Given two samples $\{x_i\}_{i=1}^{n_x} \sim P$ and $\{y_i\}_{i=1}^{n_y} \sim Q$, a simple unbiased estimator of the squared MMD in 3.7 is given by

$$\widehat{\mathrm{MMD}}_k^2(P, Q) = \frac{1}{n_x(n_x-1)} \sum_{i \neq j} k(x_i, x_j) + \frac{1}{n_y(n_y-1)} \sum_{i \neq j} k(y_i, y_j) - \frac{2}{n_x n_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} k(x_i, y_j),$$

which can be interpreted as the difference between within-sample average similarity (self-similarity excluded) and the between-sample average similarity.

HSIC. Another use of kernel embeddings is in measuring dependence between random variables taking values in some generic domains (e.g. random vectors, strings, or graphs). Recall that for any kernels $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ on the respective domains $\mathcal{X}$ and $\mathcal{Y}$, we can define $k = k_{\mathcal{X}} \otimes k_{\mathcal{Y}}$, given by

$$k\left((x, y), \left(x', y'\right)\right) = k_{\mathcal{X}}(x, x') k_{\mathcal{Y}}(y, y') \tag{3.8}$$

which is a valid kernel on the product domain $\mathcal{X} \times \mathcal{Y}$ by Lemma 12. The tensor notation signifies that the canonical feature map of k is $(x, y) \mapsto k_{\mathcal{X}}(\cdot, x) \otimes k_{\mathcal{Y}}(\cdot, y)$. Here the

feature of pair (x, y), $\varphi_{x,y} = k_{\mathcal{X}}(\cdot, x) \otimes k_{\mathcal{Y}}(\cdot, y)$ is understood as a function on $\mathcal{X} \times \mathcal{Y}$, i.e. $\varphi_{x,y}(x', y') = k_{\mathcal{X}}(x', x)k_{\mathcal{Y}}(y', y)$. The RKHS of the product kernel $k = k_{\mathcal{X}} \otimes k_{\mathcal{Y}}$ is in fact isometric to $\mathcal{H}_{k_{\mathcal{X}}} \otimes \mathcal{H}_{k_{\mathcal{Y}}}$, which can be viewed as the space of Hilbert-Schmidt operators between $\mathcal{H}_{k_{\mathcal{Y}}}$ and $\mathcal{H}_{k_{\mathcal{X}}}$. We are now ready to define an RKHS-based measure of dependence between random variables X and Y.

Definition 14. Let X and Y be random variables on domains $\mathcal{X}$ and $\mathcal{Y}$ (non-empty topological spaces). Let $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ be kernels on $\mathcal{X}$ and $\mathcal{Y}$ respectively. *Hilbert-Schmidt Independence Criterion (HSIC)* $\Xi_{k_{\mathcal{X}},k_{\mathcal{Y}}}(X, Y)$ of X and Y is the squared MMD between the joint measure P_{XY} and the product of marginals $P_X P_Y$, computed with the product kernel $k = k_{\mathcal{X}} \otimes k_{\mathcal{Y}}$, i.e.,

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

A sufficient condition for HSIC to be well defined is that both kernels $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ are bounded. The name of HSIC comes from the operator view of the RKHS $\mathcal{H}_{k_{\mathcal{X}} \otimes k_{\mathcal{Y}}}$. Namely, by repeated use of the reproducing property, it can be verified (**exercise**) that the difference between embeddings $\mu_k(P_{XY}) - \mu_k(P_X P_Y)$ can be identified with the cross-covariance operator $C_{XY} : \mathcal{H}_{k_{\mathcal{Y}}} \to \mathcal{H}_{k_{\mathcal{X}}}$ for which

$$\langle f, C_{XY} g\rangle_{\mathcal{H}_{k_{\mathcal{X}}}} = \mathrm{Cov}\left[f(X)g(Y)\right], \quad \forall f \in \mathcal{H}_{k_{\mathcal{X}}}, g \in \mathcal{H}_{k_{\mathcal{Y}}}.$$

Note that this is analogous to the finite-dimensional property $f^\top C_{XY} g = \mathrm{Cov}\left[f^\top X, g^\top Y\right]$, where X and Y are random vectors and C_{XY} is their cross-covariance matrix, i.e. $[C_{XY}]_{ij} = \mathrm{Cov}\left[X^{(i)}, Y^{(j)}\right]$. HSIC is then simply the squared Hilbert-Schmidt norm $\|C_{XY}\|^2_{HS}$ of this operator.

To obtain an estimator of the HSIC, we first express it in terms of the expectations of kernels. Starting from the definition, and expanding the Hilbert space norm into inner products:

$$\begin{aligned}\Xi_{k_{\mathcal{X}},k_{\mathcal{Y}}}(X, Y) &= \|\mathbb{E}_{XY}\left(k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y)\right) \\ &\quad - \mathbb{E}_X\left(k(\cdot, X)\right) \otimes \mathbb{E}_Y\left(k(\cdot, Y)\right)\|^2_{\mathcal{H}_k} \\ &= \langle \mathbb{E}_{XY}\left(k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y)\right), \mathbb{E}_{XY}\left(k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y)\right)\rangle_{\mathcal{H}_k} \\ &\quad + \langle \mathbb{E}_X\left(\mathbb{E}_Y\left(k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y)\right)\right), \mathbb{E}_X\left(\mathbb{E}_Y\left(k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y)\right)\right)\rangle_{\mathcal{H}_k} \\ &\quad - 2\langle \mathbb{E}_{XY}\left(k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y)\right), \mathbb{E}_X\left(\mathbb{E}_Y\left(k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y)\right)\right)\rangle_{\mathcal{H}_k} \\ &= \mathbb{E}_{XY}\left(\mathbb{E}_{X'Y'}\left(\langle k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y), k_{\mathcal{X}}(\cdot, X') \otimes k_{\mathcal{Y}}(\cdot, Y')\rangle_{\mathcal{H}_k}\right)\right) \\ &\quad + \mathbb{E}_{XX'}\left(\mathbb{E}_{YY'}\left(\langle k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y), k_{\mathcal{X}}(\cdot, X') \otimes k_{\mathcal{Y}}(\cdot, Y')\rangle_{\mathcal{H}_k}\right)\right) \\ &\quad - 2\mathbb{E}_{XY}\left(\mathbb{E}_{X'}\left(\mathbb{E}_{Y'}\left(\langle k_{\mathcal{X}}(\cdot, X) \otimes k_{\mathcal{Y}}(\cdot, Y), k_{\mathcal{X}}(\cdot, X') \otimes k_{\mathcal{Y}}(\cdot, Y')\rangle_{\mathcal{H}_k}\right)\right)\right) \\ &= \mathbb{E}_{XY}\left(\mathbb{E}_{X'Y'}\left(k_{\mathcal{X}}(X, X') k_{\mathcal{Y}}(Y, Y')\right)\right) \\ &\quad + \mathbb{E}_{XX'}\left(k_{\mathcal{X}}(X, X')\right) \mathbb{E}_{YY'}\left(k_{\mathcal{Y}}(Y, Y')\right) \\ &\quad - 2\mathbb{E}_{XY}\left(\mathbb{E}_{X'}\left(k_{\mathcal{X}}(X, X')\right) \mathbb{E}_{Y''}\left(k_{\mathcal{Y}}(Y, Y'')\right)\right)\end{aligned} \tag{3.9}$$

Here the first expectation is taken over two independent copies (X, Y), $(X', Y') \sim P_{XY}$, the second over two independent $X, X' \sim P_X$ and two independent $Y, Y' \sim P_Y$ and the third over a pair $(X, Y) \sim P_{XY}$ sampled from the joint and an independent pair $X' \sim P_X$, $Y'' \sim P_Y$. Now, given a sample $Z = \{z_i\}_{i=1}^m = \{(x_i, y_i)\}_{i=1}^m$, where each $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, we can derive an estimator of the HSIC by estimating each of the three terms in the expansion.

Denote for convenience $k_{ij} = k_{\mathcal{X}}(x_i, x_j)$ and $l_{ij} = k_{\mathcal{Y}}(y_i, y_j)$ for $i, j \in \{1, 2, ..., m\}$ and define the kernel matrices $K = (k_{ij})_{i,j=1}^m$ and $L = (l_{ij})_{i,j=1}^m$ (recall that they are symmetric and positive-definite). Following, we estimate:

$$\begin{aligned}
\widehat{\text{first term}} &= \frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m k_{ij} l_{ij} = \frac{1}{m^2} \text{tr}\,(KL) \\
\widehat{\text{second term}} &= \frac{1}{m^4} \left(\sum_{i=1}^m \sum_{j=1}^m k_{ij} \right) \left(\sum_{i=1}^m \sum_{j=1}^m l_{ij} \right) \\
&= \frac{1}{m^4} \left(1_m^T K 1_m \right) \left(1_m^T L 1_m \right) \\
\widehat{\text{third term}} &= \frac{1}{m^3} \sum_{i=1}^m \sum_{j=1}^m \sum_{q=1}^m k_{ij} l_{iq} = \frac{1}{m^3} 1_m^T K L 1_m \\
&= \frac{1}{m^3} 1_m^T L K 1_m
\end{aligned}$$

Here 1_m is the vector with m entries equal to 1. Therefore an estimator for the HSIC can be written as:

$$\begin{aligned}
\widehat{\Xi_{k_{\mathcal{X}}, k_{\mathcal{Y}}}}(X, Y) &= \frac{1}{m^2} \left(\text{tr}\,(KL) - \frac{2}{m} 1_m^T K L 1_m \; + \frac{1}{m^2} \left(1_m^T K 1_m \right) \left(1_m^T L 1_m \right) \right) \\
&= \frac{1}{m^2} \left(\text{tr}\,(KL) - \frac{1}{m} \text{tr}\left(1_m 1_m^T K L \right) - \frac{1}{m} \text{tr}\left(K 1_m 1_m^T L \right) \right. \\
&\quad \left. + \frac{1}{m^2} \text{tr}\left(1_m 1_m^T K 1_m 1_m^T L \right) \right) \\
&= \frac{1}{m^2} \text{tr}\left(\left(I - \frac{1}{m} 1_m 1_m^T \right) K \left(I - \frac{1}{m} 1_m 1_m^T \right) L \right) \\
&= \frac{1}{m^2} \text{tr}\,(KHLH)\,.
\end{aligned}$$

Here we used that $\text{tr}\,(AB) = \text{tr}\,(BA)$, $\text{tr}\,(A) = \text{tr}\left(A^T\right)$ and that any real number is equal to its own trace. We also defined

$$H \coloneqq I - \frac{1}{m} 1_m 1_m^T$$

which is the *centering matrix*. Namely, if A is any $m \times m$-matrix, AH centers the rows of A and HA centers the columns of A. Note also that H is symmetric and idempotent,

i.e. $H^2 = H$. Hence, $\operatorname{tr}((HKH)(HLH))) = \operatorname{tr}(H(KHLH)) = \operatorname{tr}(KHLHH) = \operatorname{tr}(KHLH)$.

Recall that the kernel is an inner product between features of the inputs and that inner products are bilinear. Therefore, the matrices $\widetilde{K} = HKH$ and $\widetilde{L} = HLH$ are the kernel matrices for the variables centered in feature space. We therefore arrive at the expression for the estimator:

$$\widehat{\Xi_{k_{\mathcal{X}},k_{\mathcal{Y}}}}(X,Y) = \frac{1}{m^2}\operatorname{tr}\left(\widetilde{K}\widetilde{L}\right), \tag{3.10}$$

which has an intuitive explanation of how it measures the dependence between X and Y. Namely, the function $(A,B) \to \operatorname{tr}\left(A^T B\right)$ is an inner product on the vector space of real $m \times m$ matrices. Therefore, our estimate measures the similarity between the (centered) kernel matrices, which in turn measure the "similarity patterns" between the individual observations. If there is some dependence between the X and Y, we also expect that the kernel matrices will have a similar structure and hence the inner product between them (and hence our HSIC estimator in (3.10)), will be larger.

4 Similarity Graphs and Laplacians

Given a set of data points $x_1, \ldots, x_n$ and some notion of nonnegative similarity/affinity $w_{i,j} \geq 0$ between all pairs of data points x_i and x_j, one can consider a weighted undirected *similarity* graph $G = (\{1, \ldots, n\}, \mathbf{W})$, with the *similarity/affinity matrix* $\mathbf{W} = (w_{i,j})$ corresponding to edge weights. The intuition behind similarity graph is that it should capture local neighbourhood relationships between the individual data points. Each vertex i corresponds to a data point x_i and pairs of vertices are connected by an edge if their similarity exceeds some threshold, and the edge is weighted by $w_{i,j}$. The following are some of the most common constructions of similarity graphs based on a given data set $x_1, \ldots, x_n$.

The ϵ-neighbourhood graphs. Connect all points whose pairwise distances are smaller than or equal to ϵ. Such graph is usually left unweighted, i.e. the only "metric" information kept in the graph is whether or not the points are *close* (distance at most ϵ) or far (distance greater than ϵ). Arguably, for small ϵ, as the distances between the connected points are roughly on the same scale (at most ϵ), weighting those edges does not incorporate much additional information to the graph.

The k-nearest neighbour graphs. Connect each vertex i to j if data point x_j is among the k nearest neighbours of data point x_i. However, neighbourhood relationship is not symmetric, so this construction would lead to a directed graph. There are two ways to making this graph undirected. The more common way is to simply ignore the directions of the edges, i.e. i and j are connected if x_j is among the k nearest neighbours of x_i *or* if x_i is among the k nearest neighbours of x_j. Second option (so called *mutual* nearest neighbor graph), i.e. i and j are connected if both x_j is among the k nearest neighbours of x_i *and* if x_i is among the k nearest neighbours of x_j. In both approaches, one can leave the resulting graph unweighted, or wieght the resulting edges by the similarity of their endpoints.

The fully connected graph. Here we simply compute a nonnegative similarity $w_{i,j}$ in some functional form and weight all edges by $w_{i,j}$. A non-negative kernel $\kappa(x_i, x_j)$, such as a Gaussian, $\kappa(x_i, x_j) = \exp\left(-\frac{1}{2\gamma^2}\|x_i - x_j\|^2\right)$ can be used as the graph weight w_{ij}, where the parameter γ controls the width of the neighbourhoods, similarly to the role of ϵ in the ϵ-neighbourhood graphs. Note that the use of kernel functions here is fundamentally different from that in the previous chapter (in particular, reproducing kernels need not be nonnegative).

The key quantity we will need based on the similarity graph is the *graph Laplacian*, defined below.

Definition 15. The *(unnormalized) Laplacian* of a graph $G = (\{1, \ldots, n\}, \mathbf{W})$ is an $n \times n$ matrix given by

$$\mathbf{L} = \mathbf{D} - \mathbf{W},$$

where $\mathbf{D} = \operatorname{diag}(\mathbf{W1})$ is a diagonal matrix with $\mathbf{D}_{ii} = \deg(i)$, and $\deg(i)$ denotes the *degree* of vertex i defined as

$$\deg(i) = \sum_{j=1}^{n} w_{ij}.$$

Note that the Laplacian always has the column vector $\mathbf{1}$ as an eigenvector with eigenvalue 0 (since all rows sum to zero).

Exercise 16. For all $a \in \mathbb{R}^n$

$$a^\top \mathbf{L} a = \frac{1}{2} \sum_{i,j} w_{ij} (a_i - a_j)^2 \geq 0,$$

which means that the Laplacian is a positive semi-definite matrix, and all the eigenvalues are non-negative.

The most common application of graph Laplacians is that of spectral clustering, which we describe next ([27] provides an excellent in-depth overview). However, the principles behind graph Laplacians can be applied more widely and to many other machine learning applications, such as nonlinear dimensionality reduction [2], manifold regularization [3] and ranking [11].

4.1 Spectral Clustering

4.1.1 Graph Cuts

K-means algorithm will often fail when applied to data with elongated or non-convex cluster structures. An alternative approach to clustering is to use *graph cuts* on a weighted undirected *similarity* graph $G = (\{1, \ldots, n\}, \mathbf{W})$ induced by the dataset consisting of n observations $\{x_i\}_{i=1}^n$. We wish to partition the dataset into K clusters, which can be thought of as a partition $C_1, C_2, \ldots, C_K$ of the vertex set $\{1, \ldots, n\}$. The overall *graph cut* across clusters is given by

$$\operatorname{cut}(C_1, \ldots, C_K) = \sum_{k=1}^{K} \operatorname{cut}(C_k, \bar{C}_k),$$

where $\bar{C}_k$ is the complement of C_k and $\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$ is the sum of the weights separating vertex subset A from the vertex subset B, where A and B are disjoint.

The direct cut minimization, however, results with very small cluster sizes (i.e. it would typically split a single datapoint from the rest), so one needs to balance the cuts by the cluster sizes in the partition. One approach is to consider the notion of "ratio cut"

$$\text{ratio-cut}\,(C_1, \ldots, C_K) = \sum_{k=1}^{K} \frac{\text{cut}(C_k, \bar{C}_k)}{|C_k|}.$$

Unfortunately, minimizing this criterion is computationally hard to solve. Spectral clustering algorithm uses a relaxation of the problem of minimizing the ratio cut.

4.1.2 Graph Laplacian and Ratio Cuts

The relationship between the ratio cuts and the graph Laplacian is given in the following:

Lemma 17. *For a given partition $C_1, C_2, \ldots, C_K$ define the column vectors $h_k \in \mathbb{R}^n$ as*

$$h_{k,i} = \frac{1}{\sqrt{|C_k|}} \mathbf{1}_{\{i \in C_k\}}.$$

Then

$$\textit{ratio-cut}\,(C_1, \ldots, C_K) = \sum_{k=1}^{K} h_k^\top \mathbf{L} h_k. \tag{4.1}$$

Note that the vectors h_k are orthonormal by construction. Thus, to minimize the ratio cut exactly, we can search for orthonormal vectors h_k with entries either 0 or $1/\sqrt{|C_k|}$ which minimize the RHS in (4.1). This is equivalent to integer programming so it is computationally hard. Thus, we instead look for *any collection of orthonormal vectors* h_k that minimize RHS in (4.1) – which corresponds to the eigendecomposition of the Laplacian.

If the original graph is disconnected, in addition to $\mathbf{1}$, there would be other 0-eigenvectors of $\mathbf{L}$, corresponding to the indicators of the connected components of the graph (see Theorem 25.4.1 in [18]). The idea of spectral clustering is to assume that the graph we end up with based on the dataset, while possibly not disconnected, is a "small perturbation" of a disconnected graph, and we are trying to recover connected components, i.e., clusters based on a noisy version of the true Laplacian of the underlying disconnected graph.

The algorithm proceeds by eigendecomposing $\mathbf{L}$ and taking the K eigenvectors corresponding to the K smallest eigenvalues – this gives a new "data representation"

$$\mathbf{Z} = [u_1, \ldots, u_K] \in \mathbb{R}^{n \times K}$$

on which we can apply a more conventional clustering algorithm, such as K-means.

Remark 18. A number of *normalized* graph Laplacians have also been proposed, which are based on slightly different "balancing" formulation of cuts, including the "random walk" matrix $\mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$ and $\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$.

4.2 Manifold Regularization

In semi-supervised learning, graph Laplacians are utilized to construct additional regularizers of functions $f : \mathcal{X} \to \mathcal{Y}$ mapping from inputs to the outputs. The idea is to learn the manifold shape corresponding to input distribution from unlabelled inputs and then constrain the functions using that shape. A learned function should change slowly were the inputs are dense.

Assuming we have a labelled set of examples $\{(x_i, y_i)_{i=1}^n\}$ and an unlabelled set of inputs $\{x_{n+i}\}_{i=1}^u$, we form an $(n+u) \times (n+u)$ Laplacian matrix $\mathbf{L}$ and consider the ERM with an additional *intrinsic* regularizer

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i=1}^{n+u} \sum_{j=1}^{n+u} w_{ij} (f(x_i) - f(x_j))^2$$

for the vector $\mathbf{f} = [f(x_1), \ldots, f(x_{n+u})]^\top$ of function values on all inputs, i.e. the additional regularizer penalizes large differences between function values at the neighbouring vertices i and j(where $w_{ij} > 0$). As a result, we obtain the optimization problem

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \frac{\lambda}{n} \|f\|_{\mathcal{H}}^2 + \frac{\eta}{n} \mathbf{f}^\top \mathbf{L}\mathbf{f}$$

for a given hypothesis class $\mathcal{H}$. Term $\|f\|_{\mathcal{H}}^2$ is the standard, *ambient* regularizer corresponding to the complexity of functions in the hypothesis class and λ and η, respectively, denote ambient and intrinsic regularization parameters.

If $\mathcal{H} = \mathcal{H}_k$ is an RKHS for a kernel k, a version of the representer theorem still applies, but with the solution spanned using *all inputs*, i.e. $f_\star = \sum_{i=1}^{n+u} \alpha_i k(x_i, \cdot)$, since both the empirical risk and the intrinsic regularizer term only depend on f through its evaluations at $x_1, \ldots, x_{n+u}$. In the case of a squared loss (*Laplacian Regularized Least Squares – LapRLS*), a closed form solution for dual coefficients is available (cf. SC4 2018 exam, Q3) and given by

$$\alpha = \left(\mathbf{J}^\top \mathbf{J}\mathbf{K} + \lambda \mathbf{I} + \eta \mathbf{L}\mathbf{K} \right)^{-1} \mathbf{J}^\top \mathbf{y},$$

where $\mathbf{J} = [\mathbf{I}_n \mid \mathbf{0}]$ is an $n \times (n+u)$ matrix and $\mathbf{K}$ for the kernel matrix on all $n+u$ inputs.

5 Latent Variable Models and EM algorithm

5.1 Clustering and Mixture Modelling

K-means and hierarchical clustering are non-probabilistic algorithms – based on the intuitive notions of clustering "similar" instances together and "dissimilar" instances apart. Their goal is not to model the probability of the observed data items. In contrast, *probabilistic unsupervised learning* constructs a *generative model* that describes clustering of the items. We assume that there is some latent / unobserved process that is governing the data generation - and based on the data, we will try to answer the questions about this generating process.

Mixture models assume that our dataset $\mathbf{X}$ was created by sampling iid from K distinct populations (called *mixture components*). In other words, data come from a mixture of several sources and the model for the data can be viewed as a convex combination of several distinct probability distributions, often modelled with a given parametric family.

Samples in population k can be modelled using a distribution F_{μ_k} with density $f(x|\mu_k)$, where μ_k is the *model parameter* for the k-th component. For a concrete example, consider a p-dimensional multivariate normal density with unknown mean μ_k and *known diagonal* covariance $\sigma^2 I$,

$$f(x|\mu_k) = |2\pi\sigma^2|^{-\frac{p}{2}} \exp\left(-\frac{1}{2\sigma^2}\|x-\mu_k\|_2^2\right). \tag{5.1}$$

Such model corresponds to the following generative model, whereby for each data item $i = 1, 2, \ldots, n$, we

(i) first determine the assignment variable (independently for each data item i):

$$Z_i \overset{iid}{\sim} \text{Mult}(\pi_1, \ldots, \pi_K) \qquad \text{i.e., } \mathbb{P}(Z_i = k) = \pi_k$$

where for $k = 1, \ldots, K$, $\pi_k \geq 0$, such that $\sum_{k=1}^K \pi_k = 1$, are the *mixing proportions*, additional model parameters to be inferred;

(ii) then, given the assignment $Z_i = k$ of the mixture component, $X_i = (X_i^{(1)}, \ldots, X_i^{(p)})^\top$ is sampled (independently) from the corresponding k-th component:

$$X_i|(Z_i = k) \sim f(x|\mu_k).$$

We observe $X_i = x_i$ for each i but do not observe its assignment Z_i (*latent variables*), and would like to infer the parameters $\theta = (\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K)$ as well as the latent variables.

Note that the complete log-likelihood in the model

$$\log p(\mathbf{z}, \mathbf{X}|\theta) = \log \left(\prod_{i=1}^{n} \pi_{z_i} f(x_i|\mu_{z_i}) \right) = \sum_{i=1}^{n} (\log \pi_{z_i} + \log f(x_i|\mu_{z_i})) \tag{5.2}$$

is not available as z_i is not observed. We can consider marginalising over the latent variables

$$p(\mathbf{X}|\theta) = \sum_{z_1=1}^{K} \ldots \sum_{z_n=1}^{K} \prod_{i=1}^{n} \pi_{z_i} f(x_i|\mu_{z_i}) = \prod_{i=1}^{n} \left(\sum_{k=1}^{K} \pi_k f(x_i|\mu_k) \right). \tag{5.3}$$

giving the *marginal log-likelihood* of the observations,

$$\ell(\theta) = \log p(\mathbf{X}|\theta) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k f(x_i|\mu_k).$$

However, direct maximisation is not feasible and the marginal log-likelihood will often have many local optima. Fortunately, there is a simple local marginal log-likelihood maximisation algorithm called Expectation Maximisation (EM), which we will describe in Section 5.3.

5.2 KL Divergence and Gibbs' Inequality

Before we describe the EM algorithm, we will review the notion of *Kullback-Leibler (KL) divergence* or *relative entropy* between probability distributions P and Q.

KL divergence.

- Let P and Q be two absolutely continuous probability distributions on $\mathcal{X} \subseteq \mathbb{R}^d$ with densities p and q respectively. Then the KL divergence *from Q to P* is defined as

$$D_{KL}(P \parallel Q) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx. \tag{5.4}$$

- Let P and Q be two discrete probability distributions with probability mass functions p and q respectively. Then the KL divergence *from Q to P* is defined as

$$D_{KL}(P \parallel Q) = \sum_{i} p(x_i) \log \frac{p(x_i)}{q(x_i)}. \tag{5.5}$$

In both cases, we can write

$$D_{KL}\left(P \parallel Q\right) = \mathbb{E}_p\left[\log \frac{p(X)}{q(X)}\right], \tag{5.6}$$

where $\mathbb{E}_p$ denotes that expectation is taken over p. By convexity of $f(x) = -\log(x)$ and Jensen's inequality (5.8), we have that

$$D_{KL}\left(P \parallel Q\right) = \mathbb{E}_p\left[-\log \frac{q(X)}{p(X)}\right] \geq -\log \mathbb{E}_p \frac{q(X)}{p(X)} = 0, \tag{5.7}$$

where in the last step we used that $\int_{\mathcal{X}} q(x)dx = 1$ in continuous case and $\sum_i q(x_i) = 1$ in discrete case.

Jensen's inequality. Let f be a convex function and X be a random variable. Then

$$\mathbb{E}\left[f(X)\right] \geq f\left(\mathbb{E}X\right). \tag{5.8}$$

If f is strictly convex, then equality holds if and only if X is almost surely a constant.

Thus, we conclude that KL-divergence is always non-negative. This consequence of Jensen's inequality is called *Gibbs' inequality*. Moreover, since $f(x) = -\log(x)$ is strictly convex on $x > 0$, the equality holds if and only if $p(x) = q(x)$ almost everywhere, i.e. $P = Q$. Note that in general KL-divergence is *not symmetric*: $D_{KL}\left(P \parallel Q\right) \neq D_{KL}\left(Q \parallel P\right)$.

5.3 EM Algorithm

EM algorithm is a general purpose iterative strategy for local maximisation of the likelihood under missing data/hidden variables. The method has been proposed many times for specific models– it was given its name and studied as a general framework by [9].

Let $(\mathbf{X}, \mathbf{z})$ be a pair of observed variables $\mathbf{X}$, and latent variables $\mathbf{z}$. Our probabilistic model is given by $p(\mathbf{X}, \mathbf{z}|\theta)$, but we have no access to $\mathbf{z}$. Therefore, we would like to maximise the observed data log-likelihood (marginal log-likelihood) $\ell(\theta) = \log p(\mathbf{X}|\theta) = \log \int p(\mathbf{X}, \mathbf{z}|\theta)d\mathbf{z}$ over θ. However, marginalisation of latent variables typically results in an intractable optimization problem and we need to resort to approximations.

Now, assume for a moment that we have access to another objective function $\mathcal{F}(\theta, q)$, where $q(\mathbf{z})$ is a certain distribution on latent variables $\mathbf{z}$, which we are free to choose and will call *variational distribution*. Moreover, assume that $\mathcal{F}$ satisfies

$$\mathcal{F}(\theta, q) \leq \ell(\theta) \text{ for all } \theta,\, q, \tag{5.9}$$

$$\max_q \mathcal{F}(\theta, q) = \ell(\theta), \tag{5.10}$$

i.e. $\mathcal{F}(\theta, q)$ is a *lower bound on the log-likelihood* for any variational distribution q (5.9), which also *matches the log-likelihood* at a particular choice of q (5.10).

Given these two properites, we can construct an alternating maximisation: *coordinate ascent* algorithm as follows:

Coordinate ascent on the lower bound. For $t = 1, 2 \dots$ until convergence:

$$q^{(t)} := \text{argmax}_q \mathcal{F}(\theta^{(t-1)}, q)$$
$$\theta^{(t)} := \text{argmax}_\theta \mathcal{F}(\theta, q^{(t)}).$$

Theorem 19. *Assuming* (5.9) *and* (5.10)*, coordinate ascent on the lower bound* $\mathcal{F}(\theta, q)$ *does not decrease the log likelihood* $\ell(\theta)$*.*

Proof. $\ell(\theta^{(t-1)}) = \mathcal{F}(\theta^{(t-1)}, q^{(t)}) \leq \mathcal{F}(\theta^{(t)}, q^{(t)}) \leq \mathcal{F}(\theta^{(t)}, q^{(t+1)}) = \ell(\theta^{(t)})$. Additional assumption, that $\nabla^2_\theta \mathcal{F}(\theta^{(t)}, q^{(t)})$ are negative definite with eigenvalues $< -\epsilon < 0$, implies that $\theta^{(t)} \to \theta^*$ where θ^* is a local MLE. □

But how to find such lower bound $\mathcal{F}$? It is given by the so called *variational free energy*, which we define next.

Definition 20. *Variational free energy* in a latent variable model $p(\mathbf{X}, \mathbf{z}|\theta)$ is defined as

$$\mathcal{F}(\theta, q) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{z}|\theta) - \log q(\mathbf{z})], \tag{5.11}$$

where q is any probability density/mass function over the latent variables $\mathbf{z}$.

Consider the KL divergence between $q(\mathbf{z})$ and the true conditional based on our model $p(\mathbf{z}|\mathbf{X}, \theta) = p(\mathbf{X}, \mathbf{z}|\theta)/p(\mathbf{X}|\theta)$ for the observations $\mathbf{X}$ and a fixed parameter vector θ. Since KL is non-negative,

$$\begin{aligned} 0 \leq D_{KL}\left[q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{X}, \theta)\right] &= \mathbb{E}_{\mathbf{z}\sim q} \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{X}, \theta)} \\ &= \log p(\mathbf{X}|\theta) + \mathbb{E}_{\mathbf{z}\sim q} \log \frac{q(\mathbf{z})}{p(\mathbf{X}, \mathbf{z}|\theta)}. \end{aligned}$$

Thus, we have obtained a lower bound on the marginal log-likelihood which holds true for *any parameter value* θ and *any choice of the variational distribution* q:

$$\ell(\theta) = \log p(\mathbf{X}|\theta) \geq \mathbb{E}_{\mathbf{z}\sim q} \log \frac{p(\mathbf{X}, \mathbf{z}|\theta)}{q(\mathbf{z})} = \underbrace{\mathbb{E}_{\mathbf{z}\sim q} \log p(\mathbf{X}, \mathbf{z}|\theta)}_{\text{energy}} \overbrace{-\mathbb{E}_{\mathbf{z}\sim q} \log q(\mathbf{z})}^{\text{entropy}}. \tag{5.12}$$

The right hand side in (5.12) is precisely the variational free energy - we see it decomposes in two terms. The first term is usually referred to as *energy* using the physics terminology, more precisely it is the *expected complete data log-likelihood* (if we observed $\mathbf{z}$, we would just maximise the complete data log-likelihood $\log p(\mathbf{X}, \mathbf{z}|\theta)$, but since $\mathbf{z}$

is not observed we need to integrate it out - but recall that q here is *any* distribution over latent variables). The second term is the Shannon entropy $H(q) = -\mathbb{E}_q \log q(\mathbf{z})$ of the variational distribution $q(\mathbf{z})$, and does not depend on θ (it can be thought of as the complexity penalty on q).

The inequality becomes an equality when KL divergence is zero, i.e. when $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{X}, \theta)$ which means that the optimal choice of variational distribution q for fixed parameter value θ is *the true conditional of the latent variables given the observations and that* θ.

Thus, we have proved the following lemma:

Lemma 21. *Let $\mathcal{F}$ be the variational free energy in a latent variable model $p(\mathbf{X}, \mathbf{z}|\theta)$. Then (a) $\mathcal{F}(\theta, q) \leq \ell(\theta)$ for all q and for all θ, and (b) for any θ, $\mathcal{F}(\theta, q) = \ell(\theta)$ iff $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \theta)$.*

Thus, properties (5.9) and (5.10) are satisfied and we can recast the alternating maximisation of the variational free energy into iterative updates of q (E-step, via the plug-in full conditional of $\mathbf{z}$ using the current estimate of θ) and the updates of θ (M-step, by maximising the 'energy' for the current estimate of q). Provided that both E-step and M-step can be solved exactly, EM Algorithm converges to the local maximum likelihood solution.

EM Algorithm. Initialize $\theta^{(0)}$. At time $t \geq 1$:

- E-step: Set $q^{(t)}(\mathbf{z}) = p(\mathbf{z}|\mathbf{X}, \theta^{(t-1)})$
- M-step: Set $\theta^{(t)} = \arg\max_\theta \mathbb{E}_{\mathbf{z}\sim q^{(t)}} \log p(\mathbf{X}, \mathbf{z}|\theta)$.

5.4 EM Algorithm for Mixtures

Consider again our mixture model from Section 5.1 with

$$p(\mathbf{z}, \mathbf{X}|\theta) = \prod_{i=1}^{n} \pi_{z_i} f(x_i|\mu_{z_i}).$$

Recall that our latent variables $\mathbf{z}$ are discrete (they correspond to cluster assignments) so q is a probability mass function over $\mathbf{z} := (z_i)_{i=1}^n$. Using the expression (5.2), we can

write the variational free energy as

$$\begin{aligned}\mathcal{F}(\theta, q) =& \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{z}|\theta) - \log q(\mathbf{z})] \\ =& \mathbb{E}_q\left[\left(\sum_{i=1}^{n}\sum_{k=1}^{K} \mathbf{1}(z_i = k)\left(\log \pi_k + \log f(x_i|\mu_k)\right)\right) - \log q(\mathbf{z})\right] \\ =& \sum_{\mathbf{z}} q(\mathbf{z})\left[\left(\sum_{i=1}^{n}\sum_{k=1}^{K} \mathbf{1}(z_i = k)\left(\log \pi_k + \log f(x_i|\mu_k)\right)\right) - \log q(\mathbf{z})\right] \\ =& \sum_{i=1}^{n}\sum_{k=1}^{K} q(z_i = k)\left(\log \pi_k + \log f(x_i|\mu_k)\right) + H(q).\end{aligned}$$

We will denote $Q_{ik} = q(z_i = k)$, which is called *responsibility of cluster k for data item i*.

Now, the E-step simplifies because

$$\begin{aligned}p(\mathbf{z}|\mathbf{X}, \theta) = \frac{p(\mathbf{X}, \mathbf{z}|\theta)}{p(\mathbf{X}|\theta)} = \frac{\prod_{i=1}^{n} \pi_{z_i} f(x_i|\mu_{z_i})}{\sum_{\mathbf{z}'}\prod_{i=1}^{n} \pi_{z_i'} f(x_i|\mu_{z_i'})} &= \prod_{i=1}^{n} \frac{\pi_{z_i} f(x_i|\mu_{z_i})}{\sum_k \pi_k f(x_i|\mu_k)} \\ &= \prod_{i=1}^{n} p(z_i|x_i, \theta).\end{aligned}$$

Thus, for a fixed $\theta^{(t-1)} = (\mu_1^{(t-1)}, \ldots, \mu_K^{(t-1)}, \pi_1^{(t-1)}, \ldots, \pi_K^{(t-1)})$ we can set

$$Q_{ik}^{(t)} = p(z_i = k|x_i, \theta^{(t-1)}) = \frac{\pi_k^{(t-1)} f(x_i|\mu_k^{(t-1)})}{\sum_{j=1}^{K} \pi_j^{(t-1)} f(x_i|\mu_j^{(t-1)})}. \tag{5.13}$$

Now, consider the M-step. For mixing proportions we have a constraint that $\sum_{j=1}^{K} \pi_j = 1$, so we introduce the Lagrange multiplier and obtain

$$\begin{aligned}&\nabla_{\pi_k}\left(\mathcal{F}(\theta, q) - \lambda(\textstyle\sum_{j=1}^{K} \pi_j - 1)\right) \\ &\quad = \sum_{i=1}^{n} \frac{Q_{ik}}{\pi_k} - \lambda = 0 \quad \Rightarrow \quad \pi_k \propto \sum_{i=1}^{n} Q_{ik}.\end{aligned}$$

Since

$$\sum_{k=1}^{K}\sum_{i=1}^{n} Q_{ik} = \sum_{i=1}^{n} \underbrace{\sum_{k=1}^{K} Q_{ik}}_{=1} = n,$$

the M-step update for mixing proportions is

$$\pi_k^{(t)} = \frac{\sum_{i=1}^{n} Q_{ik}^{(t)}}{n}, \tag{5.14}$$

i.e., they are simply given by the total responsibility of each cluster. Note that this update holds regardless of the form of the parametric family $f(\cdot|\mu_k)$ used for mixture components.

Setting derivative with respect to μ_k to 0, we obtain

$$\nabla_{\mu_k} \mathcal{F}(\theta, q) = \sum_{i=1}^{n} Q_{ik} \nabla_{\mu_k} \log f(x_i|\mu_k) = 0. \tag{5.15}$$

This equation can be solved quite easily for mixture of normals in (5.1), giving the M-step update

$$\mu_k^{(t)} = \frac{\sum_{i=1}^n Q_{ik}^{(t)} x_i}{\sum_{i=1}^n Q_{ik}^{(t)}}, \tag{5.16}$$

which implies that the k-th cluster mean estimate is simply a weighted average of all the data items, where the weights correspond to the responsibilities of cluster k for these points.

Put together, the EM for normal mixture model with known (fixed) covariance is very similar to K-means algorithm where cluster assignments are soft, i.e. rather than assigning each data item x_i to a single cluster at each iteration, we carry forward a responsibility vector $(Q_{i1}, \ldots, Q_{iK})$ giving probabilities of x_i belonging to each cluster. Indeed, K-means algorithm can be undestood as EM where $\sigma^2 \to 0$, such that E-step will assign exactly one entry in $(Q_{i1}, \ldots, Q_{iK})$ to one (corresponding to the nearest mean vector) and the rest to zero.

EM for Normal Mixtures (known covariance) – "Soft K-means"

1. Initialize K cluster means $\mu_1, \ldots, \mu_K$ and mixing proportions $\pi_1, \ldots, \pi_K$.
2. *Update responsibilites (E-step):* For each $i = 1, \ldots, n$, $k = 1, \ldots, K$:

$$Q_{ik} = \frac{\pi_k \exp\left(-\frac{1}{2\sigma^2}\|x_i - \mu_k\|_2^2\right)}{\sum_{j=1}^K \pi_j \exp\left(-\frac{1}{2\sigma^2}\|x_i - \mu_j\|_2^2\right)} \tag{5.17}$$

3. *Update parameters (M-step):* Set $\mu_1, \ldots, \mu_K$ and $\pi_1, \ldots, \pi_K$ and based on the new cluster responsibilities:

$$\pi_k = \frac{\sum_{i=1}^n Q_{ik}}{n}, \qquad \mu_k = \frac{\sum_{i=1}^n Q_{ik} x_i}{\sum_{i=1}^n Q_{ik}}. \tag{5.18}$$

4. Repeat steps 2-3 until convergence.
5. Return the responsibilites $\{Q_{ik}\}$ and parameters $\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K$.

In some cases, depending on the form of the parametric family $f(\cdot|\mu_k)$ the M-step update for mixtures cannot be solved exactly. In these cases, we can use *gradient ascent* algorithm *inside the M-step*:

$$\mu_k^{(r+1)} = \mu_k^{(r)} + \alpha \sum_{i=1}^{n} Q_{ik} \nabla_{\mu_k} \log f(x_i|\mu_k^{(r)}).$$

This leads to *generalized EM algorithm.*

5.5 Probabilistic PCA

So far, we have considered the application of EM to clustering, but it can be applied to latent variable models more broadly. Here, we will derive EM for *Probabilistic PCA* [26], a latent variable model for probabilistic dimensionality reduction. Just like in PCA, we try to model a collection of n p-dimensional vectors using a k-dimensional representation with $k < p$. Probabilistic PCA corresponds to the following generative model.

For each data item $i = 1, 2, \ldots, n$:

- Let Y_i be a (latent) k-dimensional normally distributed random vector with mean 0 and identity covariance:
$$Y_i \sim \mathcal{N}(0, I_k),$$
- Given Y_i, the distribution of the i-th data item is a p-dimensional normal:
$$X_i \sim \mathcal{N}(\mu + LY_i, \sigma^2 I)$$
where the parameters $\theta = (\mu, L, \sigma^2)$ correspond to a vector $\mu \in \mathbb{R}^p$, a matrix $L \in \mathbb{R}^{p \times k}$ and $\sigma^2 > 0$.

Note that unlike in clustering, the latent variables $Y_1, \ldots, Y_n$ are now continuous.

From an equivalent representation $X_i = \mu + LY_i + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I_p)$ and is independent of Y, we see that the marginal model on X_i's is

$$f(x|\theta) \;=\; \mathcal{N}\left(x; \mu, LL^\top + \sigma^2 I\right),$$

where parameters are denoted $\theta = \left(\mu, L, \sigma^2\right)$. From here it is clear that the maximum marginal likelihood estimator of μ is available directly as $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and thus, we do not require EM to estimate μ. We will henceforth assume that the data is centred, to simplify notation and remove μ from the parameters.

On the other hand, maximum marginal likelihood solution for L is unique only up to orthonormal transformations, which is why a certain form of L is usually enforced (e.g. lower-triangular, orthogonal columns). [26] shows that the MLE for PPCA has the following form. Let $\lambda_1 \geq \cdots \geq \lambda_p$ be the eigenvalues of the sample covariance and

$V_{1:k} \in \mathbb{R}^{p\times k}$ the top k eigenvectors as before. Let $Q \in \mathbb{R}^{k\times k}$ be any orthogonal matrix. Then we have:

$$\mu^{\mathrm{MLE}} = \bar{x} \qquad (\sigma^2)^{\mathrm{MLE}} = \tfrac{1}{p-k}\sum_{j=k+1}^{p}\lambda_j$$
$$L^{\mathrm{MLE}} = V_{1:k}\mathrm{diag}((\lambda_1 - (\sigma^2)^{\mathrm{MLE}})^{\frac{1}{2}}, \ldots, (\lambda_k - (\sigma^2)^{\mathrm{MLE}})^{\frac{1}{2}})Q.$$

We note that the standard PCA is recovered when $\sigma^2 \to 0$. However, the EM algorithm we derive below can be faster than eigendecomposition, can be implemented online, can handle missing data and can be extended to more complicated models. We will now proceed by deriving the EM algorithm.

E-step.

By Gaussian conditioning (*exercise*),

$$q(y_i) = p(y_i|x_i,\theta) = \mathcal{N}(y_i|b_i,R),$$

where

$$b_i = \left(L^\top L + \sigma^2 I\right)^{-1} L^\top x_i, \tag{5.19}$$
$$R = \sigma^2\left(L^\top L + \sigma^2 I\right)^{-1}. \tag{5.20}$$

M-step.

Recall that the parameters of interest are $\theta = \left(L, \sigma^2\right)$ (since the marginal maximum likelihood estimate of the mean parameter μ is directly available). We would like to maximise the variational free energy given by:

$$\mathcal{F}(\theta, q) = \mathbb{E}_{\mathbf{y}\sim q}\left[\sum_{i=1}^{n}\log p(x_i, y_i|\theta)\right] + \text{const.}$$

By ignoring terms that do not depend on θ and denoting $=_c$ to mean "equal up to a constant independent on θ"

$$\begin{aligned}\log p(x_i, y_i|\theta) &=_c -\frac{p}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(x_i - Ly_i)^\top(x_i - Ly_i)\\ &=_c -\frac{p}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\left\{x_i^\top x_i - 2x_i^\top Ly_i + y_i^\top L^\top Ly_i\right\}\\ &=_c -\frac{p}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\left\{x_i^\top x_i - 2x_i^\top Ly_i + \mathrm{Tr}\left[L^\top Ly_iy_i^\top\right]\right\}.\end{aligned}$$

Taking expectation over $q(y_i) = \mathcal{N}(y_i|b_i,R)$ gives

$$\mathbb{E}_{y_i\sim q}(\log p(x_i, y_i|\theta)) =_c -\frac{p}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\left\{x_i^\top x_i - 2x_i^\top Lb_i + \mathrm{Tr}\left[L^\top L\left(b_ib_i^\top + R\right)\right]\right\}.$$

It remains to sum over all observations to get

$$\begin{aligned}\mathcal{F}(\theta, q) &=_c -\frac{np}{2}\log\sigma^2 \\ &\quad -\frac{1}{2\sigma^2}\left\{\sum_{i=1}^{n} x_i^\top x_i - 2\sum_{i=1}^{n} x_i^\top L b_i + \operatorname{Tr}\left[L^\top L\left(\sum_{i=1}^{n} b_i b_i^\top + nR\right)\right]\right\}.\end{aligned}$$

Now, we have

$$\frac{\partial\mathcal{F}}{\partial L} = \frac{1}{\sigma^2}\left\{\sum_{i=1}^{n} x_i b_i^\top - L\left(\sum_{i=1}^{n} b_i b_i^\top + nR\right)\right\},$$

which by setting to 0 gives the update rule

$$L^{(\text{new})} = \left(\sum_{i=1}^{n} x_i b_i^\top\right)\left(\sum_{i=1}^{n} b_i b_i^\top + nR\right)^{-1}. \tag{5.21}$$

Letting $\tau = \sigma^{-2}$, we have:

$$\frac{\partial\mathcal{F}}{\partial\tau} = \frac{np}{2}\frac{1}{\tau} - \frac{1}{2}\left\{\sum_{i=1}^{n} x_i^\top x_i - 2\sum_{i=1}^{n} x_i^\top L b_i + \operatorname{Tr}\left[L^\top L\left(\sum_{i=1}^{n} b_i b_i^\top + nR\right)\right]\right\},$$

and thus

$$\left(\sigma^2\right)^{(\text{new})} = \frac{1}{np}\left\{\sum_{i=1}^{n} x_i^\top x_i - 2\sum_{i=1}^{n} x_i^\top L^{(new)} b_i + \operatorname{Tr}\left[L^{(\text{new})\top} L^{(\text{new})}\left(\sum_{i=1}^{n} b_i b_i^\top + nR\right)\right]\right\}. \tag{5.22}$$

Both Probabilistic PCA and normal mixtures are examples of linear Gaussian models, all of which have the corresponding learning algorithms based on EM. For a unifying review of these and a number of other models from the same family, including factor analysis and hidden Markov models, cf. [22].

6 Collaborative Filtering

6.1 Ratings and Recommendations

Collaborative Filtering (CF) is a collective name for a range of techniques that tackle the problem of making predictions about the preferences of a set of *users* on a set of *items*, based on the user's ratings on other items and based on the ratings of other users. Typical example concerns predicting movie preferences based on the ratings of previously watched movies – popularized by the 2006 Netflix competition.

movie \ user	Alice	Bob	Chuck	Dan	Eve
Happy Gilmore	?	2	5	1	4
Click	1	?	4	?	?
Ex Machina	?	4	?	?	2
Blade Runner	5	?	1	?	?
The Matrix	5	5	?	?	4

In a typical setup, we have a *partially observed* matrix $\mathbf{Y} \in \mathbb{R}^{n_1 \times n_2}$ where $y_{i,j}$ is the rating (e.g. between 1 and 5) of movie i by user j, assuming we have n_1 movies and n_2 users[1]. Most entries will be *missing/unknown* since most users will not have rated most movies. We will also introduce a matrix of *exposure indicators* $\mathbf{E}$ where $e_{i,j} = 1$ if the user j has rated movie i and $e_{i,j} = 0$ otherwise.

6.2 Content-Based Recommendations and Alternating Linear Regressions

In the case where additional attributes about users or about the movies *are observed*, the problem can be treated in a supervised learning fashion. Assume that for each movie i we also have access to a *feature vector* $\phi_i = [\phi_{i1}, \dots, \phi_{ik}]^\top \in \mathbb{R}^k$ (for example, ϕ_{i1} may indicate whether a movie i is a romantic comedy, ϕ_{i2} whether it is based on a comic book etc). Then we could simply formulate the problem as n_2 separate linear models[2] for each user j:

$$\min_{\psi_j} \sum_{i\,:\,e_{i,j}=1} (y_{i,j} - \phi_i^\top \psi_j)^2 + \lambda_\psi \|\psi_j\|_2^2, \quad j = 1, \dots, n_2. \tag{6.1}$$

[1] note the departure from our usual $n \times p$ convention – indeed, we will consider that both users and movies have some underlying set of variables – but that these are not necessarily observed

[2] in the typical case of integer ratings, a generalised linear model is more appropriate, as linear model can make predictions outside of the range of valid rating values, but we are keeping things simple

Here, ψ_j is the corresponding vector of coefficients in the linear model, and we have included the L_2-regularization term (which becomes important if some users have rated only a small number of movies). This model is called *content-based recommendation system* since it depends on specific features of the movies. Note that content-based recommendations are not "collaborative" in the sense that recommendations made to a user do not make use of the information across the entire user-base.

We often do not have appropriate features for movies and even if we do, it is not clear if those specific features are relevant for ratings prediction. Notice that $\psi_j = [\psi_{j1}, \dots, \psi_{jk}]^\top \in \mathbb{R}^k$ in (6.1) can be treated as a preference vector for each user j (e.g. ψ_{j1} tells us whether the user j likes romantic comedies, ψ_{j2} whether the user j likes movies based on comic books etc), so let us assume for the moment that we have access to these user preferences but not to the actual feature vectors ϕ_i for the movies. Because of the symmetry in the model, we can now infer those feature vectors, based on the preferences:

$$\min_{\phi_i} \sum_{j\,:\,e_{i,j}=1} (y_{i,j} - \phi_i^\top \psi_j)^2 + \lambda_\phi \|\phi_i\|_2^2, \quad i = 1, \dots, n_1. \tag{6.2}$$

Thus, we see that it is possible to formulate the predictions not based on features of the movies nor based on the preferences of the users (either of which may or may not be observed), but merely on the ratings matrix: we *randomly initialise* movie feature vectors ϕ_i, and then perform an iterative minimization alternating between the updates (6.1) and (6.2). This may result in features/preferences that do not have an easily understandable meaning, but are capturing salient movie/user properties that result in the ratings we observe. Moreover, by optimizing over both movie features and user preferences, predictions for each user can potentially depend on ratings of all other users (i.e. they are "collaborative").

While alternating linear regressions can be solved in closed form, due to very large numbers n_1 and n_2 of movies and users, in practice one often uses stochastic gradient descent (SGD) updates, where when we observe a new rating $y_{i,j}$, we only update the feature vector ϕ_i of movie i and the preference vector ψ_j of user j:

$$\phi_i \leftarrow (1 - \epsilon_t \lambda_\phi)\phi_i + \epsilon_t \psi_j (y_{ij} - \phi_i^\top \psi_j), \tag{6.3}$$

$$\psi_j \leftarrow (1 - \epsilon_t \lambda_\psi)\psi_j + \epsilon_t \phi_i (y_{ij} - \phi_i^\top \psi_j). \tag{6.4}$$

6.3 Low-Rank Matrix Factorization

The method of alternating linear regressions can be understood as low-rank matrix factorization of the ratings matrix $\mathbf{Y}$. Indeed, the ratings matrix is being approximated as a product of two low-rank matrices, $\Phi \in \mathbb{R}^{n_1 \times k}$, $\Psi \in \mathbb{R}^{n_2 \times k}$, where typically $k \ll \min(n_1, n_2)$, such that $\mathbf{Y} \approx \Phi\Psi^\top$. The columns $\phi^{(1)}, \dots, \phi^{(k)}$ of Φ can be interpreted as *learned attributes* of movies, whereas columns $\psi^{(1)}, \dots, \psi^{(k)}$ of Ψ can be interpreted as *learned attributes* of users.

If $\mathbf{Y}$ was fully observed then the best low-rank approximation is given by SVD, i.e. from SVD $\mathbf{Y} = UDV^\top$ we can set $\Phi = U_{1:n_1,1:k} D^{1/2}_{1:k,1:k}$ and $\Psi = V_{1:n_2,1:k} D^{1/2}_{1:k,1:k}$ and this is a solution[3] of

$$\min_{\Phi,\Psi} \underbrace{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}(y_{i,j} - \phi_i^\top \psi_j)^2}_{=\|\mathbf{Y}-\Phi\Psi^\top\|_F^2}. \tag{6.5}$$

However, as most entries in $\mathbf{Y}$ are missing, we have the optimization problem given by

$$\min_{\Phi,\Psi} \sum_{e_{i,j}=1} (y_{i,j} - \phi_i^\top \psi_j)^2. \tag{6.6}$$

This seemingly minor modification results in a difficult optimization problem which cannot be solved using standard SVD techniques. Moreover, it is typically needed to add regularization terms due to a large number of missing entries in $\mathbf{Y}$, which results exactly in the objective of the alternating linear regressions:

$$\min_{\Phi,\Psi} \sum_{e_{i,j}=1} (y_{i,j} - \phi_i^\top \psi_j)^2 + \lambda_\phi \|\Phi\|_F^2 + \lambda_\psi \|\Psi\|_F^2. \tag{6.7}$$

6.4 Probabilistic Matrix Factorization

Introduced in [23], the generative model corresponding to CF can be described as follows:

- For each movie $i = 1, \ldots, n_1$, sample independently the latent vector of features $\phi_i \sim \mathcal{N}(0, \sigma_\phi^2 I_k)$ from a k-dimensional normal distribution,
- For each user $j = 1, \ldots, n_2$, sample independently the latent vector of preferences $\psi_j \sim \mathcal{N}(0, \sigma_\psi^2 I_k)$ from a k-dimensional normal distribution,
- For each movie-user pair (i, j), sample $e_{i,j} \sim Bernoulli(p)$ independently and if $e_{i,j} = 1$, sample $y_{i,j}|\phi_i, \psi_j \sim \mathcal{N}(\phi_i^\top \psi_j, \sigma_y^2)$.

The (hyper)parameter vector here is given by $\theta = (\sigma_\phi^2, \sigma_\psi^2, \sigma_y^2)$. We can write the joint probability density of the observations and the latent variables as

[3] not unique as D can be distributed arbitrarily between Φ and Ψ - compare to the discussion of different versions of scaled biplots

$$
\begin{aligned}
p(\mathbf{Y}, \Phi, \Psi|\theta) &= \prod_{i=1}^{n_1} \frac{1}{(2\pi\sigma_\phi^2)^{k/2}} \exp\left(-\frac{1}{2\sigma_\phi^2}\|\phi_i\|_2^2\right) \\
&\quad \cdot \prod_{j=1}^{n_2} \frac{1}{(2\pi\sigma_\psi^2)^{k/2}} \exp\left(-\frac{1}{2\sigma_\psi^2}\|\psi_j\|_2^2\right) \\
&\quad \cdot \prod_{e_{i,j}=1} \frac{1}{(2\pi\sigma_y^2)^{1/2}} \exp\left(\frac{1}{2\sigma_y^2}(y_{i,j} - \phi_i^\top \psi_j)^2\right) \\
&\propto \exp\left(-\frac{1}{2\sigma_\phi^2}\|\Phi\|_F^2 - \frac{1}{2\sigma_\psi^2}\|\Psi\|_F^2 - \frac{1}{2\sigma_y^2}\sum_{e_{i,j}=1}(y_{i,j} - \phi_i^\top \psi_j)^2\right). \quad (6.8)
\end{aligned}
$$

Maximizing $\log p(\Phi, \Psi|\mathbf{Y}, \theta)$ in this model thus corresponds exactly to (6.7) with regularization parameters given by $\lambda_\phi = \sigma_y^2/\sigma_\phi^2$, $\lambda_\psi = \sigma_y^2/\sigma_\psi^2$. Since we now have a full probabilistic model, however, it is possible to consider joint inference of Φ, Ψ and θ as well as to consider more sophisticated model construction.

6.5 User-based and Item-based Collaborative Filtering

There is also a range of model-free (also called *memory-based*) methods for collaborative filtering, which are typically based on some notion of *user-user similarity* or *item-item similarity*.

User-based CF (UBCF) starts with a notion of *user-user similarity* computed based on the ratings, and then predicts ratings by *aggregating those of similar users*. Generally, it proceeds in the following three steps:

1. Assign a weight to all users with respect to similarity with the current user.
2. Select k users that have the highest similarity with the current user – commonly called the *neighbourhood*.
3. Compute a prediction using a weighted combination of the neighbours' ratings.

An example similarity κ is given by

$$
\kappa_{j,j'} = \frac{\sum_{i \in I_{jj'}}(y_{ij} - \bar{y}^j)(y_{ij'} - \bar{y}^{j'})}{\sqrt{\sum_{i \in I_{jj'}}(y_{ij} - \bar{y}^j)^2}\sqrt{\sum_{i \in I_{jj'}}(y_{ij'} - \bar{y}^{j'})^2}}, \quad (6.9)
$$

where $\bar{y}^j = \text{avg}_{i\,:\,e_{ij}=1}\{y_{ij}\}$ denotes the average rating given by user j and $I_{jj'} = \{i : e_{ij}e_{ij'} = 1\}$ is the set of movies rated by both users j and j'. Thus, this similarity is simply the Pearson correlation between the ratings columns restricted to those movies which are rated by both users. Now to make a prediction for a new movie-user pair

(i, j), we can simply aggregate predictions over the *neighbourhood* $U_k(i, j)$ of user j: the k users most similar to j who rated movie i, for example:

$$\hat{y}_{i,j} = \bar{y}^j + \frac{\sum_{j' \in U_k(i,j)} \kappa_{j,j'}(y_{i,j'} - \bar{y}^{j'})}{\sum_{j' \in U_k(i,j)} |\kappa_{j,j'}|}. \tag{6.10}$$

Item-based CF (IBCF) works analogously by aggregating predictions the user has made on similar movies. There is a large number of different variants of these algorithms, which consider different similarity measures and different aggregation strategies. See [17] for further details and references.

6.6 Biclustering

Also known as *coclustering*, this is a method for clustering both rows (items) and columns (users) in the observed data matrix. This can be a ratings matrix in CF, but can also correspond to a more general data matrix – for example, biclustering is often used in analysing gene expression data.

The intuition behind biclustering is as follows: even if the two users have a similar movie taste, it is extremely unlikely that the two have watched (and rated) the same movies (the overlap could in fact be very small). Thus, identifying similar users solely based on the similarity of their ratings may not be sufficient. Moreover, a group of users may have similar ratings across a certain group of movies, i.e. Alice and Bob both like science fiction, but have very different ratings across another type of movies, i.e. Alice also likes horrors but Bob hates them. Instead, we wish to *simultaneously* find groups of similar users and groups of similar movies.

In the biclustering method, we associate to each row i a latent indicator $r_i \in \{1, \dots, K^r\}$ and to each column j a latent indicator $c_j \in \{1, \dots, K^c\}$. Based on the cluster membership, matrix $\mathbf{Y}$ is partitioned into blocks, with y_{ij} belonging to the same block as $y_{i'j'}$ iff $(r_i, c_j) = (r_{i'}, c_{j'})$. Further, we assume that the matrix entries are i.i.d. within each block (r_i, c_j), i.e.

$$p(\mathbf{Y} \mid \mathbf{r}, \mathbf{c}, \theta) = \prod_{e_{ij}=1} p(y_{ij} | r_i, c_j, \theta) = \prod_{e_{ij}=1} p(y_{ij} | \theta_{r_i, c_j}),$$

for some parametric probability distribution $p(y_{ij}|\theta_{r_i,c_j})$. For example, we can obtain a model similar to matrix factorization by letting $\theta_{r_i,c_j} = (\phi_{r_i}, \psi_{c_j})$ for movie-group-level feature vectors $\phi_{r_i} \in \mathbb{R}^k$ and user-group-level preference vectors $\psi_{c_j} \in \mathbb{R}^k$ and $p(y_{ij}|\theta_{r_i,c_j}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_{ij} - \phi_{r_i}^\top \psi_{c_j})^2\right)$. Inference can then proceed similarly as in the EM algorithm.

7 Bayesian Learning

7.1 Bayesian Inference

So far, our treatment of probabilistic machine learning models has been *frequentist*, i.e. we used one set of tools to reason about latent variables $\mathbf{z}$ (e.g. cluster indicators in a mixture model) and another to reason about model parameters θ (e.g. parameters of mixture components defining those clusters). The generative processes we considered define the *likelihood function*: the joint distribution $p(\mathcal{D}|\theta)$ of all the observed data $\mathcal{D}$ given the model parameters θ and the learning consists in computing the maximum likelihood estimator

$$\hat{\theta} = \arg\max_{\theta \in \Theta} p(\mathcal{D}|\theta).$$

For example, in the EM algorithm (which is a frequentist method aimed at locally maximising the likelihood function), we were placing a variational distribution q on latent variables but not on θ, which was inferred using point estimates at each iteration.

In Bayesian inference, we also treat the model parameters θ as random variables and the process of learning is then computation of the *posterior distribution* $p(\theta|\mathcal{D})$. In addition to the likelihood $p(\mathcal{D}|\theta)$ specified by the generative model, one needs to also specify a prior distribution $p(\theta)$. Posterior distribution is then given by the *Bayes Theorem*:

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

where the denominator is the *marginal likelihood* or *evidence*:

$$p(\mathcal{D}) = \int_{\Theta} p(\mathcal{D}|\theta)p(\theta)d\theta.$$

All the questions about model parameters can be addressed based on the posterior. We can, for example, consider

- *Posterior mode*: $\widehat{\theta}^{\mathrm{MAP}} = \arg\max_{\theta \in \Theta} p(\theta|\mathcal{D})$ (maximum a posteriori).
- *Posterior mean*: $\widehat{\theta}^{\mathrm{mean}} = \mathbb{E}[\theta|\mathcal{D}]$.
- *Posterior variance*: $\mathrm{Var}[\theta|\mathcal{D}]$.
- *Posterior expectations of functions of parameters*: $\mathbb{E}[g(\theta)|\mathcal{D}]$ for some $g : \Theta \to \mathbb{R}^s$.

A particularly convenient choice of prior distributions are *conjugate priors* to a given likelihood function. A prior and likelihood are said to be conjugate if they result in a posterior that lies in the same parametric family as the prior.

Example: Bayesian inference on a categorical distribution.

Suppose we observe $\mathcal{D} = \{y_i\}_{i=1}^n$, with $y_i \in \{1, \ldots, K\}$, and model them as i.i.d. with the probability mass function $\pi = (\pi_1, \ldots, \pi_K)$:

$$p(\mathcal{D}|\pi) = \prod_{i=1}^{n} \pi_{y_i} = \prod_{k=1}^{K} \pi_k^{n_k}$$

with $n_k = \sum_{i=1}^n \mathbf{1}(y_i = k)$ and $\pi_k > 0$, $\sum_{k=1}^K \pi_k = 1$. The conjugate prior on π is the Dirichlet distribution $\mathrm{Dir}(\alpha_1, \ldots, \alpha_K)$ with parameters $\alpha_k > 0$, and density

$$p(\pi) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}$$

on the probability simplex $\{\pi : \pi_k > 0, \sum_{k=1}^K \pi_k = 1\}$. Since

$$p\left(\pi|\mathcal{D}\right) \quad \propto \quad \prod_{k=1}^{K} \pi_k^{n_k + \alpha_k - 1},$$

the posterior is also Dirichlet $\mathrm{Dir}(\alpha_1 + n_1, \ldots, \alpha_K + n_K)$. Posterior mean is given by

$$\widehat{\pi}_k^{\text{mean}} = \frac{\alpha_k + n_k}{\sum_{j=1}^K \alpha_j + n_j}.$$

Notice how parameters of the prior (hyperparameters) are essentially playing the role of the pseudocounts for each of the classes $1, \ldots, K$ (but they need not be integer-valued). They are reflecting prior beliefs about class proportions. For the case of two classes, this is equivalent to a $\mathrm{Beta}\left(\alpha_1, \alpha_2\right)$ prior on π_1, i.e. $p(\pi_1) = \frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_1)} \pi_1^{\alpha_1} \left(1 - \pi_1\right)^{\alpha_2}$.

7.2 Predictive distributions

How do we construct predictions based on the posterior distributions? Write the observations as $\mathcal{D} = \{x_i\}_{i=1}^n$ and assume the generative model specifies $p(x|\theta)$, e.g. a mixture model $p(x|\theta) = \sum_{k=1}^K \pi_k f\left(x|\phi_k\right)$, with $\theta = (\pi_1, \ldots, \pi_K; \phi_1, \ldots, \phi_K)$. The *posterior predictive distribution* is the conditional distribution of x_{n+1} given $\mathcal{D} = \{x_i\}_{i=1}^n$:

$$\begin{aligned} p(x_{n+1}|\mathcal{D}) &= \int_\Theta p(x_{n+1}|\theta, \mathcal{D}) p(\theta|\mathcal{D}) d\theta \\ &= \int_\Theta p(x_{n+1}|\theta) p(\theta|\mathcal{D}) d\theta. \end{aligned}$$

Thus, we predict new data by *averaging the predictive distribution over the posterior*. This is fundamentally different than predicting using a point estimate of θ, i.e. $p(x_{n+1}|\hat{\theta})$ as it takes into account the posterior uncertainty in parameters.

Example: Bayesian treatment of naïve Bayes classifer. Consider a K-class classification problem with binary input vectors, i.e. $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in \{0,1\}^p$ and $y_i \in \{1, \ldots, K\}$. *Naïve Bayes* classifier uses the following model:

$$p(y_i = k|\theta) = \pi_k, \quad p(x_i|y_i = k, \theta) = \prod_{j=1}^{p} \phi_{kj}^{x_i^{(j)}} (1 - \phi_{kj})^{1 - x_i^{(j)}},$$

i.e. it assumes that given the class labels, individual dimensions in input vectors are *independent*. The parameters of the model are collated into $\theta = ((\pi_k), (\phi_{kj}))$. It is often used in text classification where data items correspond to documents and $x_i^{(j)}$ indicates whether word j from a list of p words has appeared in document i. Class labels correspond to e.g. topics of the documents. Despite the name, naïve Bayes is often treated in a frequentist way, i.e. using maximum likelihood estimation of parameters. If we set $n_k = \sum_{i=1}^n \mathbf{1}\{y_i = k\}$, $n_{kj} = \sum_{i=1}^n \mathbf{1}(y_i = k, x_i^{(j)} = 1)$, the MLE can be written as

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad \hat{\phi}_{kj} = \frac{\sum_{i:y_i=k} x_i^{(j)}}{n_k} = \frac{n_{kj}}{n_k}.$$

But the MLEs can be problematic in some cases. For example, if the ℓ-th word did not appear in any documents labelled as class k ($n_{kl} = 0$), then $\hat{\phi}_{k\ell} = 0$. But if we then wish to compute the predictive probability once for a new document $\tilde{x}$ which contains ℓ-th word, we have:

$$\begin{aligned} &p(\tilde{y} = k|\tilde{x} \text{ with } \ell\text{-th entry equal to } 1, \hat{\theta}) \\ &\propto \hat{\pi}_k \prod_{j=1}^{p} \left(\hat{\phi}_{kj}\right)^{\tilde{x}^{(j)}} \left(1 - \hat{\phi}_{kj}\right)^{1 - \tilde{x}^{(j)}} = 0, \end{aligned}$$

since $\hat{\phi}_{k\ell} = 0$. This means that we will never attribute a new document containing word ℓ to class k (regardless of what other words in it may be!). Moreover, probability of a document under all classes can be 0 by the same reasoning.

Let us consider a Bayesian approach to the same model. We can write the likelihood as

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

For a conjugate prior, we can use $\mathrm{Dir}((\alpha_k)_{k=1}^K)$ for π, and $\mathrm{Beta}(a, b)$ for ϕ_{kj} independently. Now, because the likelihood factorises, the posterior distribution over π and (ϕ_{kj}) also factorises, and posterior for π is $\mathrm{Dir}((\alpha_k + n_k)_{k=1}^K)$, and for ϕ_{kj} is $\mathrm{Beta}(a + n_{kj}, b + n_k - n_{kj})$. If we want to predict a label $\tilde{y}$ for a new document $\tilde{x}$, we obtain

$$p(\tilde{x}, \tilde{y} = k|\mathcal{D}) = p(\tilde{y} = k|\mathcal{D}) p(\tilde{x}|\tilde{y} = k, \mathcal{D})$$

with

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

and the predicted class is

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

Compared to the MLE plug-in predictions, pseudocounts help to "regularise" probabilities away from the extreme values.

7.3 Laplace Approximation

Bayesian approach to learning is conceptually very elegant, but the posterior distributions are intractable in almost all interesting cases, and we therefore need to resort to various approximations. One of the techniques for approximation of intractable posterior distributions is the *Laplace* or *saddlepoint approximation*. The idea is to simply approximate the posterior distribution $p(\theta|\mathcal{D})$ with a (multivariate) Gaussian distribution. Given the ease of manipulating Gaussians, this is a convenient choice, since the various posterior expectations and predictive distributions will be easier to calculate when we have Gaussian approximate posteriors.

Consider for simplicity the case where parameter θ is a scalar and assume that posterior mode $\widehat{\theta}^{\text{MAP}}$ is available. Often, the posterior mode can be found even if the normalising constant $p(\mathcal{D})$ is intractable since it suffices to maximise $p(\theta|\mathcal{D}) \propto p(\theta, \mathcal{D}) = p(\mathcal{D}|\theta)p(\theta)$ using a numerical method. Then, we can use a Taylor expansion of $\log p(\theta|\mathcal{D})$ around the posterior mode $\widehat{\theta}^{\text{MAP}}$:

$$\begin{aligned}\log p(\theta|\mathcal{D}) &= \log p(\widehat{\theta}^{\text{MAP}}|\mathcal{D}) + \left.\frac{\partial \log p(\theta|\mathcal{D})}{\partial \theta}\right|_{\theta=\widehat{\theta}^{\text{MAP}}} \left(\theta - \widehat{\theta}^{\text{MAP}}\right) \\ &\quad + \left.\frac{\partial^2 \log p(\theta|\mathcal{D})}{\partial \theta^2}\right|_{\theta=\widehat{\theta}^{\text{MAP}}} \frac{\left(\theta - \widehat{\theta}^{\text{MAP}}\right)^2}{2} + \mathcal{O}\left(\left(\theta - \widehat{\theta}^{\text{MAP}}\right)^3\right).\end{aligned}$$

By ignoring the third and higher order terms and noticing that the the first derivative at the mode must be zero, we have an approximation:

$$\log p(\theta|\mathcal{D}) \approx \log p(\widehat{\theta}^{\text{MAP}}|\mathcal{D}) - \frac{\tau}{2}\left(\theta - \widehat{\theta}^{\text{MAP}}\right)^2, \tag{7.1}$$

where we write $\tau = -\frac{\partial^2 \log p(\theta|\mathcal{D})}{\partial \theta^2} \geq 0$. But recall that $\log \mathcal{N}\left(\theta|\mu, \sigma^2\right) = \log\left(\left(2\pi\sigma^2\right)^{-1/2}\right) - \frac{1}{2\sigma^2}\left(\theta - \mu\right)^2$, so this second order Taylor approximation has exactly the form of a normal

log-density with mean $\mu = \widehat{\theta}^{\mathrm{MAP}}$ and variance $\sigma^2 = \tau^{-1}$ so we can approximate the posterior with $\mathcal{N}\left(\widehat{\theta}^{\mathrm{MAP}}, \left(-\frac{\partial^2 \log p(\theta|\mathcal{D})}{\partial \theta^2}\right)^{-1}\right)$.

This idea easily extends to multivariate densities. In particular, the Laplace approximation of $p(\theta|\mathcal{D})$ is a multivariate Gaussian $\mathcal{N}\left(\widehat{\theta}^{\mathrm{MAP}}, \Sigma\right)$, where *the inverse covariance matrix is given by the negative Hessian of the log-posterior* evaluated at the posterior mode:

$$\Sigma^{-1} = -\frac{\partial^2 \log p(\theta|\mathcal{D})}{\partial \theta \partial \theta^\top}\bigg|_{\theta=\widehat{\theta}^{\mathrm{MAP}}}.$$

Since $\log p(\theta|\mathcal{D})$ agrees with $\log p(\theta, \mathcal{D})$ up to a constant, they have the same derivatives, so often we work with the *energy function* $J(\theta) = -\log p(\theta, \mathcal{D})$, which is the negative logarithm of the unnormalised posterior. Then we can write

$$\Sigma^{-1} = \frac{\partial^2 J(\theta)}{\partial \theta \partial \theta^\top}\bigg|_{\theta=\widehat{\theta}^{\mathrm{MAP}}}.$$

7.4 Bayesian Model Selection

Consider a situation where we do not have one Bayesian model but several. Each model $\mathcal{M}$ has a set of parameters $\theta_{\mathcal{M}}$, likelihood $p(\mathcal{D}|\theta_{\mathcal{M}})$ and the prior distribution $p(\theta_{\mathcal{M}})$. Within each model, the posterior distribution is

$$p(\theta_{\mathcal{M}}|\mathcal{D}, \mathcal{M}) = \frac{p(\mathcal{D}|\theta_{\mathcal{M}}, \mathcal{M}) p(\theta_{\mathcal{M}}|\mathcal{M})}{p(\mathcal{D}|\mathcal{M})}$$

where the normalising constant is the marginal probability of the data under model $\mathcal{M}$ (*Bayesian model evidence*):

$$p(\mathcal{D}|\mathcal{M}) = \int_\Theta p(\mathcal{D}|\theta_{\mathcal{M}}, \mathcal{M}) p(\theta_{\mathcal{M}}|\mathcal{M}) d\theta$$

In Bayesian model selection, one compares models using their *Bayes factors* $\frac{p(\mathcal{D}|\mathcal{M})}{p(\mathcal{D}|\mathcal{M}')}$.

Considering Bayesian model evidence can be interpreted as a Bayesian version of *Occam's Razor*: of two explanations adequate to explain the same set of observations, the simpler should be preferred. Namely, note that the model evidence $p(\mathcal{D}|\mathcal{M})$ is the probability that a set of randomly selected parameter values (under the prior) inside the model would generate dataset $\mathcal{D}$. In that case, models that are *too simple* are unlikely to generate the observed dataset. On the other hand, models that are *too complex* can generate many possible datasets, so again, they are unlikely to generate that particular dataset at random.

8 Variational Methods

One of the workhorses of Bayesian machine learning are *variational approximations*, which turn posterior inference in intractable Bayesian models into optimization. We have seen that Bayesian model selection proceeds by optimizing (maximizing) the model evidence. While model evidence is almost always intractable, using the same principles as in the EM algorithm (Gibbs inequality), lower bounds may be available which can be optimized instead.

8.1 ELBO

Assume that we are taking a Bayesian approach to inference in a latent variable model $p(\mathbf{X}, \mathbf{z}|\theta)$ with observations $\mathbf{X}$, latent variables $\mathbf{z}$ and parameters θ. Now, because we are using a Bayesian model, our treatment of latent variables and model parameters is exactly the same. We can now consider some joint distribution $q(\mathbf{z}, \theta)$ of latent variables and parameters, called variational distribution (similarly to EM, but note that EM was not allowed to place a distribution over θ!). We claim that the quantity

$$\mathcal{F}(q) = \mathbb{E}_q \left[\log p(\mathbf{X}, \mathbf{z}, \theta)\right] + H(q) \tag{8.1}$$

is a lower bound on log-evidence $\log p(\mathbf{X})$. Namely, we can write

$$\begin{aligned} \mathcal{F}(q) &= \mathbb{E}_q \left[\log p(\mathbf{X}, \mathbf{z}, \theta)\right] - \mathbb{E}_q[\log q(\mathbf{z}, \theta)] \\ &= \mathbb{E}_q \left[\log p(\mathbf{z}, \theta|\mathbf{X})\right] + \log p(\mathbf{X}) - \mathbb{E}_q[\log q(\mathbf{z}, \theta)] \\ &= -\mathrm{KL}\left(q(\mathbf{z}, \theta)||p(\mathbf{z}, \theta|\mathbf{X})\right) + \log p(\mathbf{X}), \end{aligned}$$

which is by Gibbs inequality maximised (and equal to log-evidence) when KL is zero, i.e. when $q(\mathbf{z}, \theta) = p(\mathbf{z}, \theta|\mathbf{X})$. Thus, for any variational distribution q, $\mathcal{F}(q) \leq \log p(x)$. Expression 8.1 is called the *evidence lower bound (ELBO)*.

To reason about all the unknowns in the model, we would simply need to compute the joint posterior $p(\mathbf{z}, \theta|\mathbf{X})$, but this is almost always intractable. Hence, the variational Bayesian inference *approximates* the posterior by starting with a family $\mathcal{Q}$ of tractable variational distributions $q(\mathbf{z}, \theta)$ (e.g. $q(\mathbf{z}, \theta|\nu)$ where ν are the *variational parameters*), and aims to minimize the divergence $\mathrm{KL}\left(q(\mathbf{z}, \theta)||p(\mathbf{z}, \theta|\mathbf{X})\right)$ over $\mathcal{Q}$ or, equivalently, maximise the ELBO, i.e. find the tightest lower bound on the log-evidence.

In a nutshell, variational Bayes projects the (intractable) posterior $p(\mathbf{z}, \theta|\mathbf{X})$ onto a tractable family $\mathcal{Q}$ with respect to the KL divergence $\mathrm{KL}\left(q(\mathbf{z}, \theta)||p(\mathbf{z}, \theta|\mathbf{X})\right)$. Alternative divergences are possible in this context. In particular, since KL is not symmetric, minimization of the "reverse" divergence $\mathrm{KL}\left(p(\mathbf{z}, \theta|\mathbf{X})||q(\mathbf{z}, \theta)\right)$ results in a different family of approximate Bayesian methods, known as Expectation Propagation (EP).

8.2 Bayesian EM and Mean-Field Variational Family

Variational approximation requires specifying the variational family $\mathcal{Q}$. The complexity of $\mathcal{Q}$ determines the difficulty of the optimization; it is more difficult to optimize over a large family $\mathcal{Q}$ than over a simpler, smaller one. Consider family $\mathcal{Q}$ of variational distributions which factorize across the latents and the parameters: $q(\mathbf{z}, \theta) = q_{\mathbf{Z}}(\mathbf{z})\, q_{\Theta}(\theta)$. For a fixed q_{Θ}, we can solve for $q_{\mathbf{Z}}$ which maximises ELBO (*exercise):*

$$q_{\mathbf{Z}}(\mathbf{z}) \propto \exp\left(\int \log p(\mathbf{X}, \mathbf{z}, \theta) q_{\Theta}(\theta)\, d\theta\right),$$

and by symmetry, for a fixed $q_{\mathbf{Z}}$, we can solve for q_{Θ} which maximises ELBO:

$$q_{\Theta}(\theta) \propto \exp\left(\int \log p(\mathbf{X}, \mathbf{z}, \theta) q_{\mathbf{Z}}(\mathbf{z})\, d\mathbf{z}\right).$$

Now, one can formulate an algorithm similar to EM, which alternates between optimising $q_{\mathbf{Z}}$ and q_{Θ}, such that each iteration increases ELBO and thus decreases the KL divergence from the posterior.

We notice the symmetry between $\mathbf{z}$ and θ. Indeed, the distinction between parameters and latent variables disappears in Bayesian modelling, as all unobserved quantities in the model are treated in the same way and our goal is to approximate their posterior distribution. In the rest of this section, we will drop θ from the notation and treat them as a part of the set of all unobserved quantities $\mathbf{z}$.

The further simplification we often make in Variational Bayes is to focus on the *mean-field variational family* where the variational distribution fully factorizes

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j),$$

i.e. all latent variables are mutually independent and each latent z_j is governed by its own variational factor q_j. Note that there could be a mix between categorical and continuous latents, each having the appropriate factor q_j. Also, z_j itself need not be a univariate latent – see an example with LDA below. Using the mean-field family implies that we will not be able to capture any posterior correlations between the latent variables z_j and $z_{j'}$ for $j \neq j'$ and that the best we can hope for is a rich representations of the posterior marginals.

The iterative procedure similar to Bayesian EM can now be applied to each individual factor, giving rise to the algorithm 8.1 called *Coordinate Ascent Variational Inference (CAVI)*, closely related to Gibbs sampling, i.e. it also uses full conditionals $p(z_j|\mathbf{z}_{-j}, \mathbf{x}) \propto p(\mathbf{z}, \mathbf{x})$, where we denoted $\mathbf{z}_{-j} = [z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_n]$.

8.3 Complete conditionals in the exponential family

When the complete conditionals $p(z_j|\mathbf{z}_{-j}, \mathbf{x})$ belong to an exponential family of distributions, i.e. they are given by

Algorithm 8.1 Coordinate Ascent Variational Inference (CAVI)

Input: a model $p(\mathbf{z}, \mathbf{x})$, dataset $\mathbf{x}$
Output: a variational posterior $q(\mathbf{z})$

while the ELBO has not converged **do**

- **for** $j = 1, \ldots, m$
  - $q_j(z_j) \propto \exp\left[\mathbb{E}_{\mathbf{z}_{-j} \sim q} \log p\left(z_j | \mathbf{z}_{-j}, \mathbf{x}\right)\right]$
- $\text{ELBO}(q) = \mathbb{E}_{\mathbf{z} \sim q}\left[\log p(\mathbf{x}, \mathbf{z})\right] + H(q)$

return $q\left(\mathbf{z}\right) = \prod_{j=1}^{m} q_j\left(z_j\right)$

$$p(z_j | \mathbf{z}_{-j}, \mathbf{x}) = h\left(z_j\right) \exp\left[\eta_j\left(\mathbf{z}_{-j}, \mathbf{x}\right)^\top z_j - A\left(\eta_j\left(\mathbf{z}_{-j}, \mathbf{x}\right)\right)\right],$$

a particular convenient form of CAVI is available, i.e. the updates in Algorithm 8.1 are available in closed form. Above, we assume z_j is already transformed to its appropriate sufficient statistic, $h\left(\cdot\right)$ is a base measure, $A\left(\cdot\right)$ is the log-normalizer and η_j are the natural parameters (which depend on the conditioning set). Now, the CAVI update reads

$$\begin{aligned} q_j(z_j) &\propto \exp\left[\mathbb{E}_{-j} \log p\left(z_j | \mathbf{z}_{-j}, \mathbf{x}\right)\right] \\ &= \exp\left[\log h\left(z_j\right) + \left\{\mathbb{E}_{-j} \eta_j\left(\mathbf{z}_{-j}, \mathbf{x}\right)\right\}^\top z_j - \mathbb{E}_{-j} A\left(\eta_j\left(\mathbf{z}_{-j}, \mathbf{x}\right)\right)\right] \\ &\propto h\left(z_j\right) \exp\left[\left\{\mathbb{E}_{-j} \eta_j\left(\mathbf{z}_{-j}, \mathbf{x}\right)\right\}^\top z_j\right] \end{aligned}$$

and thus, the variational factors are in the same exponential family as the complete conditionals with natural parameter being the expected natural parameter of the complete conditional

$$\nu_j = \mathbb{E}_{-j} \eta_j\left(\mathbf{z}_{-j}, \mathbf{x}\right).$$

This setup describes many models, including Bayesian Gaussian mixtures, Dirichlet process mixtures, matrix factorization, multilevel regression and latent Dirichlet allocation, giving thus one classical overarching CAVI algorithm with closed-form updates for many instances of Variational Bayes.

While the distinction between parameters and latent variables disappears in Bayesian modelling, there is still a relevant distinction in terms of where in the model hierarchy these unobserved quantities appear. In that respect, we can differentiate *global latent vector* β which is associated to all observations and the *local latents* $\{z_i\}_{i=1}^n$ each of which is associated to an individual observation x_i, such that the observation x_i is conditionally independent of $\{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n\}$ given β and z_i. The joint density is then

$$p(\beta, \mathbf{z}, \mathbf{x}) = p(\beta) \prod_{i=1}^{n} p(z_i, x_i|\beta).$$

The normal mixtures are an example of this, where the mixture parameters are the global latents, while cluster assignments are the local latents. The impact of such hierarchy is that not all updates in Algorithm 8.1 need to be performed sequentially. Multiple levels of hierarchy are also possible.

We next study a concrete example of this in topic modelling, Latent Dirichlet Allocation [5].

8.4 Example: Topic Modelling

Topic models are a class of probabilistic models of text that lead to parsimonious representations of hidden thematic structure of a collection of documents. A popular approach to topic modelling is Latent Dirichlet Allocation (LDA[1]) [5].

Latent Dirichlet Allocation

LDA captures the intuition that a text document typically exhibits multiple topics and blends them in a particular way. In LDA, each topic is modelled as a probability distribution over words and each document as a mixture of corpus-wide topics (i.e. it can be identified with a distribution over topics). Each observed word in a document is then treated as a draw from the mixture, i.e. it belongs to one of the topics (mixture components). Mixture proportions are thus unique for each document, i.e. they are local latents, but mixture components are shared across the whole collection - they are global latents. This setting is also called a *mixed membership model*. The goal of the LDA is to find the posterior

$$p(\text{topics,proportions,assignments}|\text{observed words})$$

Note that a corpus of text to be analyzed may consist of millions of documents, thus having possibly billions of latent variables.

LDA posits the following conditionally conjugate model, Let K be the number of topics and V the size of the vocabulary.

1. For each topic in $k = 1, \ldots, K$,
   a) Draw a distribution over V words $\beta_k \sim \text{Dir}_V(\eta)$
2. For each document in $d = 1, \ldots, D$,
   a) Draw a vector of topic proportions $\theta_d \sim \text{Dir}_K(\alpha)$
   b) For each word in $n = 1, \ldots, N_d$,

[1] Do not confuse it with Linear Discriminant Analysis which shares the acronym - these two models are not related.

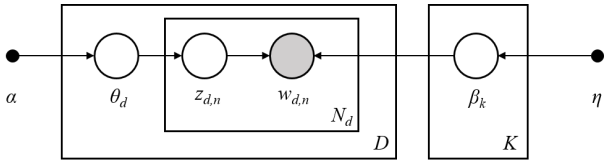

Figure 8.1: Graphical model representation of LDA. Plates represent replication, for example there are D documents each having a topic proportion vector θ_d

i. Draw a topic assignment $z_{dn} \sim \text{Mult}(\theta_d)$, i.e. $p(z_{dn} = k|\theta_d) = \theta_{dk}$

ii. Draw a word $w_{dn} \sim \text{Mult}(\beta_{z_{dn}})$, i.e. $p(w_{dn} = v|\beta, z) = \beta_{z_{dn}v}$

Thus, we can write the joint distribution as

$$\begin{aligned} p(\beta, \theta, z, w) &= \prod_{k=1}^{K} p(\beta_k; \eta) \prod_{d=1}^{D} \left\{ p(\theta_d; \alpha) \prod_{n=1}^{N_d} p(z_{dn}|\theta_d)\, p(w_{dn}|\beta, z) \right\} \\ &= \frac{1}{B(\eta)^K B(\alpha)^D} \prod_{k=1}^{K} \prod_{v=1}^{V} \beta_{kv}^{\eta_v - 1} \prod_{d=1}^{D} \left\{ \prod_{k=1}^{K} \theta_{dk}^{\alpha_k - 1} \prod_{n=1}^{N_d} \theta_{d,z_{dn}} \beta_{z_{dn},w_{dn}} \right\} \end{aligned} \tag{8.2}$$

The model has the following latents: β (topics), θ (proportions), and z (assignments). Note that there are also hyperparameter vectors $\eta \in \mathbb{R}_+^V$ and $\alpha \in \mathbb{R}_+^K$ in the Dirichlet priors - these are assumed fixed. Data are the observed words $\{w_{dn}\}$. There will be some abuse of notation here - we denote by w_{dn} both the appropriate draw from vocabulary $\{1, \ldots, V\}$ - to be used for indexing, and its "one-hot" encoding i.e. a binary V-vector with $w_{dn}[v] = 1$ if $w_{dn} = v$ and zero otherwise. We will write $w_{dn}[\cdot]$ in the case of the latter. Similarly, we denote by z_{dn} both the appropriate topic assignment from $\{1, \ldots, K\}$ and its "one-hot" encoding i.e. a binary K-vector with $z_{dn}[k] = 1$ if $z_{dn} = k$ and zero otherwise. We will write $z_{dn}[\cdot]$ in the case of the latter.

We will use a mean-field family of the form

$$q(\beta, \theta, z) = \prod_{k=1}^{K} q(\beta_k; \lambda_k) \prod_{d=1}^{D} \left\{ q(\theta_d; \gamma_d) \prod_{n=1}^{N_d} q(z_{dn}; \phi_{dn}) \right\}.$$

The complete conditionals are proportional to the joint distribution in (8.2):

1. Complete conditional on the topic assignment is a multinomial with

$$p(z_{dn} = k|\theta_d, \beta, w_d) \propto \theta_{dk} \beta_{k,w_{dn}} = \exp(\log \theta_{dk} + \log \beta_{k,w_{dn}}). \tag{8.3}$$

Thus, for the variational approximation we also use a multinomial but with a "free parameter" ϕ_{dn}, where we denote $\phi_{dn}[k] = q(z_{dn} = k)$, i.e. ϕ_{dn} is simply a probability mass function over K topics.

2. Complete conditional on the topic proportions depends only on the assignments and is given by
$$p(\theta_d|z_d) = \mathrm{Dir}_K\left(\theta_d; \alpha + \sum_{n=1}^{N_d} z_{dn}[\cdot]\right). \tag{8.4}$$
For the variational approximation we also use Dirichlet, with parameter vector $\gamma_d \in \mathbb{R}_+^K$.

3. Complete conditional on the topics is
$$p(\beta_k|z,w) = \mathrm{Dir}_V\left(\beta_k; \eta + \sum_{d=1}^{D}\sum_{n=1}^{N_d} z_{dn}[k]\, w_{dn}[\cdot]\right). \tag{8.5}$$
For the variational approximation we also use Dirichlet, with parameter vector $\lambda_k \in \mathbb{R}_+^V$.

With these full conditionals we can derive the CAVI updates in the LDA model. We will need the following fact about the Dirichlet distribution given here without proof.

Fact 22. *If $\pi \sim Dir_L(\alpha)$, then*
$$\mathbb{E}[\log \pi_j] = \psi(\alpha_j) - \psi\left(\sum_{\ell=1}^{L}\alpha_\ell\right),$$
where $\psi(u) = \frac{\Gamma'(u)}{\Gamma(u)} = \int_0^\infty \left(\frac{e^{-t}}{t} - \frac{e^{-ut}}{1-e^{-t}}\right) dt$ is the digamma function.

Now we can obtain the closed-form updates for each set of the latents.

Proposition 23. *CAVI updates in the LDA model are given by*

1. $\phi_{dn}[k] \propto \exp\left(\psi(\gamma_{dk}) + \psi(\lambda_{k,w_{dn}}) - \psi\left(\sum_{v=1}^{V}\lambda_{k,v}\right)\right)$,
2. $\gamma_d = \alpha + \sum_{n=1}^{N_d}\phi_{dn}$,
3. $\lambda_k = \eta + \sum_{d=1}^{D}\sum_{n=1}^{N_d}\phi_{dn}[k]\, w_{dn}[\cdot]$,

where ψ is the digamma function.

Proof. Steps (2) and (3) directly follow from the exponential family properties of Dirichlet distribution and are left for exercise. For (1), we make use of Fact 22 and write
$$\begin{aligned}
\phi_{dn}[k] &\propto \exp\left(\mathbb{E}_{\theta_d,\beta\sim q}\log p(z_{dn}=k|\theta_d,\beta,w_d)\right)\\
&\propto \exp\left(\mathbb{E}_{\theta_d\sim q}\log\theta_{dk} + \mathbb{E}_{\beta_k\sim q}\log\beta_{k,w_{dn}}\right)\\
&\propto \exp\left(\psi(\gamma_{dk}) - \psi\left(\sum_{\ell=1}^{K}\gamma_{d\ell}\right) + \psi(\lambda_{k,w_{dn}}) - \psi\left(\sum_{v=1}^{V}\lambda_{k,v}\right)\right)\\
&\propto \exp\left(\psi(\gamma_{dk}) + \psi(\lambda_{k,w_{dn}}) - \psi\left(\sum_{v=1}^{V}\lambda_{k,v}\right)\right),
\end{aligned}$$
as required. □

8.5 Variational Autoencoders

A popular use of variational methods is in the context of training probabilistic deep generative models known as Variational Autoencoders (VAEs) [14, 21]. An autoencoder consists of a pair of neural networks which are jointly trained to attempt to approximately copy inputs at the outputs while passing them through a lower-dimensional representation. They consist of (a) an *encoder* (also called *recognition model*) $q_\phi(z|x)$, a conditional probability distribution of codes $z \in \mathbb{R}^{d_z}$, given a high-dimensional input $x \in \mathbb{R}^{d_x}$, typically parametrized by a neural network with weights ϕ, and (b) a decoder (also called *generative model*) $p_\theta(x|z)$, a conditional probability distribution of outputs $x \in \mathbb{R}^{d_x}$, given a code $z \in \mathbb{R}^{d_z}$, also parametrized by a neural network with weights θ. Typically, $d_z \ll d_x$. In VAEs, codes are interpreted as local latent variables in a model which places a prior $p(z)$ and has a likelihood specified by the decoder $p_\theta(x|z)$. Thus, encoder is essentially a variational approximation to the intractable posterior of latent codes and training VAEs proceeds by jointly fitting the model parameters θ and the variational parameters ϕ using stochasting gradient descent [2]. After a VAE has been trained, new examples can be generated by drawing new codes z from the prior and passing them through the decoder network (e.g. if VAE was trained on a database of images, it will learn to generate new images).

The encoder typically uses a normal distribution with mean and covariance parameterized by a neural network, i.e.

$$q_\phi(z|x) = \mathcal{N}\left(z|\mu_\phi(x), \Sigma_\phi(x)\right), \tag{8.6}$$

but many other options are possible, including augmenting VAE posterior approximations by transforming drawn samples through mappings (so called *flows*) with additional trainable parameters to achieve richer variational families.

VAE ELBO. Consider different ways in which we can write the ELBO for a single observation x:

$$\begin{aligned}
\mathcal{L}(x,\theta,\phi) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z)\right] + H\left(q_\phi(\cdot|x)\right) \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(z)}{q_\phi(z|x)}\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \\
&= -KL\left(q_\phi(z|x)\,||p(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right].
\end{aligned} \tag{8.7}$$

The final expresssion is particularly convenient as for a normal variational posterior in (8.6) and a normal prior $p(z)$, the KL divergence term is available in closed form. The prior is typically just $p(z) = \mathcal{N}(z|0,I)$, and the KL term thus reads

$$KL\left(q_\phi(z|x)\,||p(z)\right) = \frac{1}{2}\left[\mu_\phi(x)^\top \mu_\phi(x) + \operatorname{tr}\left(\Sigma_\phi(x)\right) - \log\det\left(\Sigma_\phi(x)\right) - d_z\right].$$

[2] note that θ and ϕ are typically not treated in a Bayesian way, i.e. only the codes are latents

In the most common case of a mean field approximation where $\Sigma_\phi(x)$ is a diagonal matrix with $[\Sigma_\phi(x)]_{jj} = \sigma^2_{\phi,j}(x)$, this further simplifies to

$$KL\left(q_\phi(z|x)\,||p(z)\right) = \frac{1}{2}\sum_{j=1}^{d_z}\left[\mu^2_{\phi,j}(x) + \sigma^2_{\phi,j}(x) - \log\left(\sigma^2_{\phi,j}(x)\right) - 1\right].$$

Note that in this case the encoder network produces $2d_z$ outputs $\{\mu_{\phi,j}(x)\}_{j=1}^{d_z}$ and $\left\{\sigma^2_{\phi,j}(x)\right\}_{j=1}^{d_z}$, based on the d_x-dimensional input x, using weights ϕ.

To obtain the ELBO objective on the whole set of observations $\{x_i\}_{i=1}^n$,we average the individual terms in (8.7), i.e.

$$\mathcal{L}(\theta,\phi) = \frac{1}{n}\sum_{i=1}^{n}\left\{\mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right] - KL\left(q_\phi(z|x_i)\,||p(z)\right)\right\}. \tag{8.8}$$

Assuming that $\{(x_i,z_i)\}_{i=1}^n$ are drawn i.i.d. from $p_\theta(x,z)$, $\mathcal{L}(\theta,\phi)$ is a lower bound on the (scaled) model evidence $\frac{1}{n}\log p_\theta\left(\{x_i\}_{i=1}^n\right) = \frac{1}{n}\sum_{i=1}^n \log p_\theta(x_i)$, since $\mathcal{L}(x_i,\theta,\phi) \leq \log p_\theta(x_i)$, for all i. We wish to proceed with stochastic gradient descent to jointly maximize (8.8) with respect to θ and ϕ using minibatches of observations x_i at the time in order to compute unbiased estimators of the gradients of ELBO. We note, however, that the terms $\mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right]$ are generally not tractable, which we address next.

Reparametrization trick. In order to optimize the ELBO objective over θ and ϕ, it remains to estimate the terms $\mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right]$. Getting a good Monte Carlo estimator would require drawing many samples of codes for each obervation x_i, which is prohibitive. Thus, one is tempted to proceed with drawing a single $z_i \sim q_\phi(z|x_i)$ and estimating

$$\hat{\mathbb{E}}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right] = \log p_\theta(x_i|z_i).$$

However, this is clearly problematic, as explicit dependence of this estimator on the variational parameters ϕ has been lost and we cannot compute its gradients with respect to ϕ. There is a simple solution, however, known as the "reparametrization trick". For example, in the case of a normal variational posterior, since a draw $z_i \sim \mathcal{N}\left(z|\mu_\phi(x),\Sigma_\phi(x)\right)$ can be written as $z_i = \mu_\phi(x) + \Sigma_\phi^{1/2}(x)\,\epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0,I)$, we can rewrite

$$\mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right] = \mathbb{E}_\epsilon\left[\log p_\theta\left(x_i|\mu_\phi(x) + \Sigma_\phi^{1/2}(x)\,\epsilon\right)\right],$$

and use estimator of the form $\log p_\theta\left(x_i|\mu_\phi(x) + \Sigma_\phi^{1/2}(x)\,\epsilon_i\right)$, based on a single draw $\epsilon_i \sim \mathcal{N}(0,I)$. Now, it is possible to compute gradients $\nabla_\theta \log p_\theta\left(x_i|\mu_\phi(x) + \Sigma_\phi^{1/2}(x)\,\epsilon_i\right)$ and $\nabla_\phi \log p_\theta\left(x_i|\mu_\phi(x) + \Sigma_\phi^{1/2}(x)\,\epsilon_i\right)$ with respect to both θ and ϕ, and, importantly, they are unbiased estimators of $\nabla_\theta \mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right]$ and $\nabla_\phi \mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right]$, respectively. For more general variational posteriors, a similar reparametrization approach is possible whenever we can sample from $q_\phi(z|x_i)$ by computing some function

$h(\epsilon, \phi, x_i)$, where ϵ is a draw from a known distribution with no further parameters to be learned.

VAEs and unbiased estimators of $p_\theta(x)$. Assume that we have access to some stricly positive unbiased estimator $\hat{p}_\theta(x)$ of $p_\theta(x)$, with

$$\int \hat{p}_\theta(x)\, q_{\theta,\phi}(u|x)\, du = p_\theta(x),$$

where $u \sim q_{\theta,\phi}(\cdot|x)$ denotes all random variables used to compute the estimator $\hat{p}_\theta(x) = \hat{p}_\theta(x; u, \phi)$, with ϕ denoting any additional parameters of the sampling distribution. Jensen's inequality then tells us that

$$\int \log \hat{p}_\theta(x)\, q_{\theta,\phi}(u|x)\, du \quad \leq \quad \log \int \hat{p}_\theta(x)\, q_{\theta,\phi}(u|x)\, du \leq \log p_\theta(x).$$

Thus, the term

$$\mathcal{L}(x, \theta, \phi) := \int \log \hat{p}_\theta(x; u, \phi)\, q_{\theta,\phi}(u|x)\, du \tag{8.9}$$

is a lower bound on evidence for any choice of an unbiased estimator $\hat{p}_\theta(x; u, \phi)$. VAE framework we described above corresponds to simply setting $u = z$ and the estimator is of the form $\hat{p}_\theta(x) = p_\theta(x, z)/q_\phi(z|x)$. However, one can consider other types of estimators. For example, Importance Weighted Autoencoder (IWAE) [8] uses an estimator based on s importance samples $u = \{z_j\}_{j=1}^s$, with $z_j \sim q_\phi(\cdot|x)$

$$\hat{p}_\theta(x) = \frac{1}{s} \sum_{j=1}^{s} \frac{p_\theta(x, z_j)}{q_\phi(z_j|x)}.$$

While the lower bound in (8.9) may not be tractable, it suffices to only compute unbiased estimators of its gradients with respect to parameters θ and ϕ. Similarly to before, if $\mathbb{E}_{q_\phi(u|x)}[\log \hat{p}_\theta(x; u, \phi)] = \mathbb{E}_\epsilon[\log \hat{p}_\theta(x; h(\epsilon, \phi, x), \phi)]$, for some known distribution of ϵ and a given function h, then we can simply use $\nabla_\theta \log \hat{p}_\theta(x; h(\epsilon, \phi, x), \phi)$ and $\nabla_\phi \log \hat{p}_\theta(x; h(\epsilon, \phi, x), \phi)$.

9 Gaussian Processes

9.1 Different views of regression

Regression with least squares loss $L(y, f(x)) = (y - f(x))^2$ implies that we are fitting the *conditional mean* function $f^*(x) = \mathbb{E}[Y|X = x]$. This loss also corresponds to the probabilistic model where y_i is a noisy version of the underlying function f evaluated at input x_i:

$$y_i|f(x_i) \sim \mathcal{N}(f(x_i), \sigma^2), \quad \text{independently for} \quad i = 1, \ldots, n. \tag{9.1}$$

There are different ways to model the class of functions f.

- *Frequentist Parametric* approach: model f as f_θ for some parameter vector θ. Fit θ by ML / ERM with squared loss (**linear regression**).
- *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 (**kernel ridge regression**)
- *Bayesian Parametric* approach: model f as f_θ for some parameter vector θ. Put a prior on θ and compute a posterior $p(\theta|\mathcal{D})$ (**Bayesian linear regression**).
- *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 $p(f|\mathcal{D})$ (**Gaussian Process regression**).

9.2 Gaussian Process Regression

Gaussian processes (GPs) are a widely used class of models that allow us to place a prior distribution directly on the space of functions rather than on parameters in a particular family of functions. This prior can then be converted into a posterior distribution once we have seen some data. One can think of a Gaussian process as an infinite-dimensional generalisation of a multivariate normal distribution. Namely, given an *index set* $\mathcal{X}$, a collection of random variables $\{A_x\}_{x \in \mathcal{X}}$ is said to be a Gaussian process if and only if for every finite set of indices $x_1, \ldots, x_n$, vector $[A_{x_1}, \ldots, A_{x_n}]^\top$ has a multivariate normal distribution on $\mathbb{R}^n$. Thus, to any Gaussian process, we can associate a random function $f \colon \mathcal{X} \to \mathbb{R}$ by setting $f(x) = A_x$, for all $x \in \mathcal{X}$. Gaussian process is fully specified by its mean and covariance functions, i.e.

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)], \\ k(x, x') &= \mathbb{E}\left[(f(x) - m(x))\left(f(x') - m(x')\right)\right], \end{aligned}$$

where expectations are taken over f (x and x' are fixed elements in the index set $\mathcal{X}$). This means that for any finite set $x_1, \ldots, x_n$, $\mathbf{f} = [f(x_1), \ldots, f(x_n)]^\top \in \mathbb{R}^n$ has a distribution $\mathcal{N}(\mathbf{m}, \mathbf{K})$, where $\mathbf{m}_i = m(x_i)$ and $\mathbf{K}$ is the covariance matrix given by $\mathbf{K}_{ij} = k(x_i, x_j)$. We will typically assume that the mean function $m(x)$ is zero under the Gaussian process prior. If we know before seeing any data that the distribution of the function evaluations should be centered around some other mean, we could easily include that into the model. Equivalently, we could also subtract that known mean from the data and just use the zero mean model. If we are looking at the data to estimate the mean function, then often the zero mean GP suffices – in fact, structural information about mean functions (constant, linear) can be included into the choice of the covariance function (*exercises*). Covariance functions $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ obviously has to be positive definite so they are essentially equivalent to the kernel functions we have seen before. In fact, there is a rich connection between RKHS methods and Gaussian processes, an example of which we will discuss below.

9.2.1 Gaussian Conditioning and Regression Model

The convenience of manipulating multivariate normal distributions carries over to Gaussian processes. Let us review the rules for Gaussian conditioning, which are key to Gaussian process regression.

Gaussian Conditioning. Let $\mathbf{z} \sim \mathcal{N}(\mu, \Sigma)$ be a multivariate normal random vector and let us split its dimensions into two parts, i.e.

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}. \tag{9.2}$$

Note that $\Sigma_{21} = \Sigma_{12}^\top$ due to symmetry of covariance matrices. Then the conditional density of $\mathbf{z}_2$ given $\mathbf{z}_1$ is also normal and given by

$$p(\mathbf{z}_2 | \mathbf{z}_1) = \mathcal{N}(\mathbf{z}_2;\, \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{z}_1 - \mu_1),\, \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}). \tag{9.3}$$

For a given set of inputs $\mathbf{x} = \{x_i\}_{i=1}^n$, we denote the vector of evaluations of f by $\mathbf{f} = [f(x_1), \ldots, f(x_n)]^\top \in \mathbb{R}^n$ and the vector of observed outputs by $\mathbf{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$. Note that since we treat f as a random function, $\mathbf{f}$ is a random n-dimensional vector. The Gaussian process regression model, assuming likelihood function in (9.1), is then given by

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

where $\mathbf{K}$ is the covariance (kernel) matrix given by $\mathbf{K}_{ij} = k(x_i, x_j)$. But because both the prior and the likelihood are normal this simply means that $\mathbf{f}$ and $\mathbf{y}$ are *jointly normal*

with

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}\mathbf{K} & \mathbf{K}\\ \mathbf{K} & \mathbf{K}+\sigma^2 I\end{bmatrix}\right). \tag{9.4}$$

For example, to find the cross-covariance between $\mathbf{f}$ and $\mathbf{y}$ note that

$$\mathbb{E}\left[\mathbf{f}\mathbf{y}^\top\right] = \mathbb{E}\left[\mathbf{f}(\mathbf{f}+\sigma\epsilon)^\top\right] = \mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right] + \sigma\mathbb{E}\left[\mathbf{f}\epsilon^\top\right] = \mathbf{K}, \tag{9.5}$$

where $\epsilon \sim \mathcal{N}(0, I)$ is independent of $\mathbf{f}$. Now, we can simply apply the Gaussian conditioning to find the posterior distribution

$$\mathbf{f}|\mathbf{y} \sim \mathcal{N}(\mathbf{K}(\mathbf{K}+\sigma^2 I)^{-1}\mathbf{y}, \mathbf{K} - \mathbf{K}(\mathbf{K}+\sigma^2 I)^{-1}\mathbf{K}).$$

This gives as the posterior distribution of the evaluations of the unknown function at the set of inputs where we have observed noisy evaluations $\mathbf{y}$.

9.2.2 Posterior Predictive Distribution

But we can continue with this formalism further and construct the *posterior predictive distribution*. Suppose $\mathbf{x}' = \{x'_j\}_{j=1}^m$ is a test set. We can extend our model to include the function values $\mathbf{f}' = [f(x'_1), \ldots, f(x'_m)]^\top \in \mathbb{R}^m$ at the test set. The prior can now be extended to include $\mathbf{f}'$ (recall that our prior was on the whole function – not on its values at specific locations!), so that the model reads:

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

where $(\mathbf{K}_{\mathbf{xx}})_{ij} = k(x_i, x_j)$, $(\mathbf{K}_{\mathbf{x}'\mathbf{x}'})_{ij} = k(x'_i, x'_j)$, $\mathbf{K}_{\mathbf{xx}'}$ is an $n \times m$ matrix with (i,j)-th entry $k(x_i, x'_j)$ and $\mathbf{K}_{\mathbf{x}'\mathbf{x}} = \mathbf{K}_{\mathbf{xx}'}^\top$ We are now making use of the joint normality of $\mathbf{f}'$ and $\mathbf{y}$:

$$\begin{bmatrix}\mathbf{f}'\\ \mathbf{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{\mathbf{x}'\mathbf{x}'} & \mathbf{K}_{\mathbf{x}'\mathbf{x}}\\ \mathbf{K}_{\mathbf{xx}'} & \mathbf{K}_{\mathbf{xx}}+\sigma^2 I\end{bmatrix}\right) \tag{9.6}$$

and from Gaussian conditioning rules again, we can read off the posterior predictive distribution as

$$\mathbf{f}'|\mathbf{y} \sim \mathcal{N}\left(\mathbf{K}_{\mathbf{x}'\mathbf{x}}(\mathbf{K}_{\mathbf{xx}}+\sigma^2 I)^{-1}\mathbf{y}, \mathbf{K}_{\mathbf{x}'\mathbf{x}'} - \mathbf{K}_{\mathbf{x}'\mathbf{x}}(\mathbf{K}_{\mathbf{xx}}+\sigma^2 I)^{-1}\mathbf{K}_{\mathbf{xx}'}\right). \tag{9.7}$$

Thus, we also have a closed form expression for the posterior distribution of the evaluations of the unknown function at any collection of inputs in $\mathcal{X}$. While this follows directly from the joint normality and Gaussian conditioning rules, it is instructive to notice that we could have arrived at the posterior predictive by integrating $p(\mathbf{f}'|\mathbf{f})$ through the posterior $p(\mathbf{f}|\mathbf{y})$, i.e.

$$p(\mathbf{f}'|\mathbf{y}) = \int p(\mathbf{f}'|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f}. \tag{9.8}$$

This follows from $\int \mathcal{N}(a|Bc, D)\mathcal{N}(c|e, F)dc = \mathcal{N}(a|Be, D+BFB^\top)$ (*exercises*). Namely, even if we no longer have the Gaussian observation model (9.1) and $\mathbf{y}$ and $\mathbf{f}$ are no longer jointly normal, we can still use (9.8) to reason about the posterior predictive distribution.

9.2.3 Kernel Ridge Regression vs Gaussian Process Regression

If kernel ridge regression (KRR) uses the same kernel as the covariance function in Gaussian process regression (GPR) and moreover, if the regularisation parameter λ in KRR is the same as the noise variance σ^2 in GPR, KRR estimate of the function coincides with the GPR posterior mean. Indeed, recall that in KRR we are solving empirical risk minimisation

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \sigma^2 \|f\|^2_{\mathcal{H}_k},$$

and are fitting a function of the form $f(x) = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i)$. Closed form solution is given by $\alpha = \left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1} \mathbf{y}$. But then if we wish to predict function values at a new set $\mathbf{x}' = \{x'_j\}_{j=1}^{m}$ of input vectors, we have

$$f(x'_j) = \sum_{i=1}^{n} \alpha_i k\left(x'_j, x_i\right) = \left[k(x'_j, x_1), \ldots, k(x'_j, x_n)\right] (\mathbf{K_{xx}} + \sigma^2 I)^{-1} \mathbf{y},$$

and $\left[k(x'_j, x_1), \ldots, k(x'_j, x_n)\right]$ is the j-th row of $\mathbf{K_{x'x}}$, so this is the same as the mean in (9.7). Note that GPR also gives predictive variance, a measure of uncertainty, which can be important when making predictions far away from the input data. There are other important differences between the two approaches: KRR is frequentist, while GPR is Bayesian, and thus the hyperparameters are fitted in different ways. KRR typically uses cross-validation and grid search, while GPR, as we discuss next, uses maximum marginal likelihood or a fully Bayesian treatment with hyperparameters integrated out.

9.3 Hyperparameter Selection

Probabilistic model given by Gaussian processes allows principled selection of hyperparameters in the model (parameters of the kernel function and the noise variance in the likelihood (9.1)) using *maximum marginal likelihood*.

Marginal likelihood of the hyperparameter vector $\theta = (\nu, \sigma^2)$ which would generally include kernel parameters ν as well as the standard deviation σ^2 of the noise in the observation model, is given by

$$p(\mathbf{y}|\theta) = \int p(\mathbf{y}|\mathbf{f}, \theta) p(\mathbf{f}|\theta) d\mathbf{f} = \mathcal{N}\left(\mathbf{y}; 0, \mathbf{K}_\nu + \sigma^2 I\right).$$

We will introduce the shorthand $\mathbf{K}_{\theta+} = \mathbf{K}_\nu + \sigma^2 I$. Thus, we can write the marginal log-likelihood as

$$\log p(\mathbf{y}|\theta) = -\frac{1}{2} \log |\mathbf{K}_{\theta+}| - \frac{1}{2} \mathbf{y}^\top \mathbf{K}_{\theta+}^{-1} \mathbf{y} - \frac{n}{2} \log(2\pi). \tag{9.9}$$

In general, marginal log-likelihood is a nonconvex function of the parameter vector θ and it can have multiple maxima - thus we typically resort to numerical optimisation

methods, such as gradient ascent. The derivative with respect to θ_i (*exercise*) has the form

$$\frac{\partial}{\partial \theta_i} \log p(\mathbf{y}|\theta) = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{K}_{\theta+}^{-1}\frac{\partial \mathbf{K}_{\theta+}}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{y}^\top \mathbf{K}_{\theta+}^{-1}\frac{\partial \mathbf{K}_{\theta+}}{\partial \theta_i}\mathbf{K}_{\theta+}^{-1}\mathbf{y}. \tag{9.10}$$

Some common kernel choices in this context involve *Automatic Relevance Determination (ARD)* kernel

$$k(x, x') = \tau^2 \exp\left(-\sum_{j=1}^{p} \frac{(x^{(j)} - x'^{(j)})^2}{\eta_j^2}\right), \tag{9.11}$$

which has a global scale parameter τ as well as one *bandwidth* parameter η_j per covariate dimension j. If in the hyperparameter selection, very large values of η_j are selected, this essentially means that the dimension j is switched off (does not contribute to the kernel function). This is very useful in applications where it is likely that not all dimensions will be relevant.

In addition to maximum marginal likelihood, we can also perform full Bayesian inference for hyperparameters. Namely, we could start with a prior $p(\theta)$ on θ and draw samples from the posterior

$$p(\theta|\mathbf{y}) \quad \propto \quad p(\theta)p(\mathbf{y}|\theta) = p(\theta)\int p(\mathbf{y}|\mathbf{f}, \theta)p(\mathbf{f}|\theta)d\mathbf{f}.$$

This means that we can integrate uncertainty over hyperparameters into predictions as well, and approximate (integral is typically not available in closed form)

$$p(\mathbf{f}'|\mathbf{y}) = \int p(\mathbf{f}'|\mathbf{y}, \theta)p(\theta|\mathbf{y})d\theta.$$

9.4 Gaussian Processes for Classification

In Bayesian classification problems, we are interested in modelling the posterior probabilities of the categorical response variable given a set of training examples and a new input vector. These probabilities must lie in the interval $(0, 1)$ while a Gaussian process models functions that have output on the entire real axis. Thus, it is necessary to adapt Gaussian processes by transforming their outputs using an appropriate nonlinear activation/link function. Consider the binary classification model with classes -1 and $+1$, using the logistic sigmoid:

$$p(y_i = +1|f(x_i)) = \sigma(f(x_i)) = \frac{1}{1 + e^{-f(x_i)}}. \tag{9.12}$$

This non-Gaussian form of the likelihood function, however, renders exact posterior inference intractable and approximate methods are needed. There are a number of approximate schemes that can be used but we will focus here on Laplace approximation. We know that

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

Thus, we can compute the gradient

$$\frac{\partial \log p(\mathbf{f}|\mathbf{y})}{\partial \mathbf{f}} = -\mathbf{K}^{-1}\mathbf{f} + \mathbf{g}_\mathbf{f}, \tag{9.13}$$

where the gradient of the likelihood is $\mathbf{g}_\mathbf{f} = \frac{\partial \log p(\mathbf{y}|\mathbf{f})}{\partial \mathbf{f}}$ with $[\mathbf{g}_\mathbf{f}]_i = \frac{\partial \log p(\mathbf{y}|\mathbf{f})}{\partial f_i} = \sigma(-y_i f(x_i))y_i$. The Hessian is given by

$$\frac{\partial^2 \log p(\mathbf{f}|\mathbf{y})}{\partial \mathbf{f}\partial \mathbf{f}^\top} = -\mathbf{K}^{-1} - \mathbf{D}_\mathbf{f}, \tag{9.14}$$

where $\mathbf{D}_\mathbf{f} = -\frac{\partial^2 \log p(\mathbf{y}|\mathbf{f})}{\partial \mathbf{f}\partial \mathbf{f}^\top}$ is the negative Hessian of the log-likelihood, which is an $n \times n$ diagonal matrix[1] with $(\mathbf{D}_\mathbf{f})_{ii} = \sigma(f(x_i))\sigma(-f(x_i))$. The overall Hessian of the log-posterior is negative definite, since $\mathbf{K}^{-1}$ is positive definite and $(\mathbf{D}_\mathbf{f})_{ii} \geq 0$. Thus, there is a unique posterior mode. Note also that $\mathbf{D}_\mathbf{f}$ depends on $\mathbf{f} = [f_1, \ldots, f_n]^\top$ but not on the labels $\mathbf{y}$. We can now employ numerical optimisation (gradient ascent or Newton-Raphson method) to find the posterior mode $\hat{\mathbf{f}}^{\text{MAP}}$ and approximate the posterior $p(\mathbf{f}|\mathbf{y})$ with a normal distribution:

$$\tilde{p}(\mathbf{f}|\mathbf{y}) = \mathcal{N}\left(\mathbf{f} \,\middle|\, \hat{\mathbf{f}}^{\text{MAP}}, \left(\mathbf{K}^{-1} + \mathbf{D}_{\hat{\mathbf{f}}^{\text{MAP}}}\right)^{-1}\right).$$

Note that this can be rewritten as

$$\tilde{p}(\mathbf{f}|\mathbf{y}) = \mathcal{N}\left(\mathbf{f} \,\middle|\, \hat{\mathbf{f}}^{\text{MAP}}, \mathbf{K} - \mathbf{K}\left(\mathbf{K} + \mathbf{D}^{-1}_{\hat{\mathbf{f}}^{\text{MAP}}}\right)^{-1}\mathbf{K}\right),$$

using the Woodbury identity[2] $\left(\mathbf{K}^{-1} + \mathbf{D}\right)^{-1} = \mathbf{K} - \mathbf{K}\left(\mathbf{K} + \mathbf{D}^{-1}\right)^{-1}\mathbf{K}$ for invertible matrices $\mathbf{K}$ and $\mathbf{D}$.

We can use the Laplace approximation further to construct an approximation of the predictive posterior at a test set $\mathbf{x}' = \{x'_j\}_{j=1}^m$, writing

$$\tilde{p}(\mathbf{f}'|\mathbf{y}) = \int p(\mathbf{f}'|\mathbf{f})\tilde{p}(\mathbf{f}|\mathbf{y})d\mathbf{f}, \tag{9.15}$$

[1] Note that $\frac{\partial^2 \log p(\mathbf{y}|\mathbf{f})}{\partial \mathbf{f}\partial \mathbf{f}^\top}$ is a diagonal matrix in any GP model regardless of the form of the likelihood function as long as it factorizes across observations, i.e. $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^n p(y_i|f_i)$ with $(\mathbf{D}_\mathbf{f})_{ii} = -\frac{\partial^2 \log p(y_i|f_i)}{\partial f_i^2}$. In addition, if $\log p(y|f)$ is concave in f, $(\mathbf{D}_\mathbf{f})_{ii} \geq 0$.

[2] Woodbury matrix identity or matrix inversion lemma in its general form is $(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}VA^{-1}$ for matrices A,U,C,V of conformable sizes.

which can now be solved in the closed form since $p(\mathbf{f}'|\mathbf{f})$ is also normal,

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

giving

$$\tilde{p}\left(\mathbf{f}'|\mathbf{y}\right) = \mathcal{N}\left(\mathbf{f}' \,|\, \mathbf{K}_{\mathbf{x}'\mathbf{x}}\mathbf{K}_{\mathbf{x}\mathbf{x}}^{-1}\hat{\mathbf{f}}^{\mathrm{MAP}}, \mathbf{K}_{\mathbf{x}'\mathbf{x}'} - \mathbf{K}_{\mathbf{x}'\mathbf{x}}\left(\mathbf{K}_{\mathbf{x}\mathbf{x}} + \mathbf{D}_{\hat{\mathbf{f}}^{\mathrm{MAP}}}^{-1}\right)^{-1}\mathbf{K}_{\mathbf{x}\mathbf{x}'}\right). \tag{9.16}$$

Let us compare this to the predictive distribution $p\left(\mathbf{f}'|\hat{\mathbf{f}}^{\mathrm{MAP}}\right)$ based on simply plugging in the point estimate $\hat{\mathbf{f}}^{\mathrm{MAP}}$ at the training points, which is

$$p\left(\mathbf{f}'|\hat{\mathbf{f}}^{\mathrm{MAP}}\right) = \mathcal{N}\left(\mathbf{f}' \,|\, \mathbf{K}_{\mathbf{x}'\mathbf{x}}\mathbf{K}_{\mathbf{x}\mathbf{x}}^{-1}\hat{\mathbf{f}}^{\mathrm{MAP}}, \mathbf{K}_{\mathbf{x}'\mathbf{x}'} - \mathbf{K}_{\mathbf{x}'\mathbf{x}}\mathbf{K}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{K}_{\mathbf{x}\mathbf{x}'}\right).$$

Two distributions have the same mean but the plug-in predictive underestimates the variance. To see this, note that for any test point $x_\star$, the predictive variances are

$$\begin{aligned}
\mathrm{var}\left[f\left(x_\star\right)|\mathbf{y}\right] &= k_{\star\star} - \mathbf{k}_{\star\mathbf{x}}\left(\mathbf{K}_{\mathbf{x}\mathbf{x}} + \mathbf{D}_{\hat{\mathbf{f}}^{\mathrm{MAP}}}^{-1}\right)^{-1}\mathbf{k}_{\mathbf{x}\star}, \\
\mathrm{var}\left[f\left(x_\star\right)|\hat{\mathbf{f}}^{\mathrm{MAP}}\right] &= k_{\star\star} - \mathbf{k}_{\star\mathbf{x}}\mathbf{K}_{\mathbf{x}\mathbf{x}}^{-1}\mathbf{k}_{\mathbf{x}\star},
\end{aligned}$$

and positive-definiteness of $\mathbf{D}_{\hat{\mathbf{f}}^{\mathrm{MAP}}}^{-1}$ implies $\left(\mathbf{K}_{\mathbf{x}\mathbf{x}} + \mathbf{D}_{\hat{\mathbf{f}}^{\mathbf{MAP}}}^{-1}\right)^{-1} \preceq \mathbf{K}_{\mathbf{x}\mathbf{x}}^{-1}$, whereby $\mathrm{var}\left[f\left(x_\star\right)|\mathbf{y}\right] \geq \mathrm{var}\left[f\left(x_\star\right)|\hat{\mathbf{f}}^{\mathrm{MAP}}\right]$.

An alternative to the logistic link is the probit model, i.e.

$$p(y_i = +1|f(x_i)) = \Phi(f(x_i)), \tag{9.17}$$

where $\Phi(z) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z} e^{-t^2/2}dt$ is the standard normal cdf. Derivations proceed similarly by considering the gradient and Hessian of the log-posterior

$$\log p(\mathbf{f}|\mathbf{y}) = \mathrm{const} - \frac{1}{2}\mathbf{f}^\top\mathbf{K}^{-1}\mathbf{f} + \sum_{i=1}^{n}\log\Phi(y_i f(x_i)).$$

Thus, it suffices to replace

$$\begin{aligned}
\left(\mathbf{g}_{\mathbf{f}}\right)_i &= \frac{y_i\phi\left(f_i\right)}{\Phi\left(y_i f_i\right)}, \\
\left(\mathbf{D}_{\mathbf{f}}\right)_{ii} &= \frac{\phi\left(f_i\right)^2}{\Phi\left(y_i f_i\right)^2} + \frac{y_i f_i\phi\left(f_i\right)}{\Phi\left(y_i f_i\right)}
\end{aligned}$$

in (9.13) and (9.14), where $\phi(z) = \Phi'\left(z\right)$ is the standard normal pdf. The overall Hessian is again negative definite as a consequence of the log-concavity of Φ.

9.5 Numerically stable implementation

Kernel matrix $\mathbf{K}$ can in practice have eigenvalues close to zero and thus be numerically unstable to invert. Fortunately, the direct inversion of $\mathbf{K}$ can be avoided. Consider, for example, the Newton iteration for finding the MAP given by

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

with the last expression involving only the inverse of $\mathbf{K} + \mathbf{D}_\mathbf{f}^{-1}$ and not of $\mathbf{K}$. A recommended implementation [20, Section 3.4.3] is to consider the matrix

$$\mathbf{B} = \mathbf{D}_\mathbf{f}^{1/2}\left(\mathbf{K} + \mathbf{D}_\mathbf{f}^{-1}\right)\mathbf{D}_\mathbf{f}^{1/2} = \mathbf{D}_\mathbf{f}^{1/2}\mathbf{K}\mathbf{D}_\mathbf{f}^{1/2} + I$$

which is guaranteed to be well conditioned for most kernel functions since its eigenvalues are between 1 and $1 + n \max_{i,j} \mathbf{K}_{ij}/4$ and perform its Cholesky decomposition $\mathbf{B} = \mathbf{L}\mathbf{L}^\top$. The Newton update can then be implemented as

$$\mathbf{f}^{new} = \mathbf{K}\left(I - \mathbf{D}_\mathbf{f}^{1/2}\mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{D}_\mathbf{f}^{1/2}\mathbf{K}\right)\left(\mathbf{D}_\mathbf{f}\mathbf{f} + \mathbf{g}_\mathbf{f}\right).$$

9.6 Large-Scale Kernel Approximations

Gaussian processes and kernel methods require computational cost that scales at least as $O(n^2)$ and often as $O(n^3)$ in the number of observations n (due to the need to compute, store and invert the $n \times n$ kernel matrix $\mathbf{K}$). This is the price we pay for having a nonparametric model, i.e. for performing the computation in terms of the dual coefficients. For large datasets, e.g. where $n \sim 10^5$, this becomes a prohibitive computational cost and memory requirement, however. Many methods have been proposed to deal with this issue, here we will overview the basic approaches based on the reduced-rank approximation of $\mathbf{K}_{\mathbf{xx}}$ - see Chapter 8 of [20] for an in-depth overview.

9.6.1 Nyström method

GP regression and kernel ridge regression both require inversion of the matrix $\mathbf{K}_{\mathbf{xx}} + \sigma^2 I$. Let us assume for the moment that $\mathbf{K}_{\mathbf{xx}}$ can be approximated by a rank m matrix, with $m \ll n$, i.e. $\mathbf{K}_{\mathbf{xx}} \approx QQ^\top$, where Q is an $n \times m$ matrix. Then we can apply the matrix inversion lemma and write

$$\left(QQ^\top + \sigma^2 I\right)^{-1} = \sigma^{-2}I - \sigma^{-2}Q\left(\sigma^2 I + Q^\top Q\right)^{-1}Q^\top, \tag{9.18}$$

such that the inversion of an $n \times n$ matrix has been transformed into an inversion of an $m \times m$ matrix. However, in order to derive the optimal reduced-rank approximation to $\mathbf{K_{xx}}$, we need to perform the eigendecomposition of $\mathbf{K_{xx}}$ which is itself a costly operation, requiring $O(n^3)$ computation. Instead, an often used approach is the *Nyström approximation:*

$$\tilde{\mathbf{K}}_{\mathbf{xx}} = \mathbf{K_{xz}} \mathbf{K_{zz}^{-1}} \mathbf{K_{zx}},$$

where $\{z_j\}_{j=1}^m$ is a collection of a small number of inputs in $\mathcal{X}$ (which could be a subset of the training set, but could also be some auxiliary pseudo-inputs) called *inducing variables* or *landmark points*, and we denoted as usual $(\mathbf{K_{zz}})_{ij} = k(z_i, z_j)$, $(\mathbf{K_{xz}})_{ij} = k(x_i, z_j)$ and $\mathbf{K_{zx}} = \mathbf{K_{xz}^\top}$. Now, we can set $Q = \mathbf{K_{xz}} \mathbf{K_{zz}^{-1/2}}$ and apply the formula (9.18). Note that this is equivalent to using a finite-dimensional feature map $\phi : x \mapsto \mathbf{K_{zz}^{-1}} \left[k(z_1, x), \ldots, k(z_m, x)\right]^\top$ and an *approximate kernel*:

$$\tilde{k}(x, x') = \phi(x)^\top \phi(x').$$

9.6.2 Random Fourier Features

Another popular method within the frequentist kernel methods are random Fourier features (RFF) of [19]. The idea behind RFF is to use Bochner's representation of *translation-invariant* kernels on $\mathbb{R}^p$, i.e. if a real-valued kernel $k(x, x')$ depends only on the difference $x - x'$, then it can be written as

$$\begin{aligned} k(x, x') &= \int_{\mathbb{R}^p} \exp\left(i\omega^\top (x - x')\right) d\Lambda(\omega) \\ &= \int_{\mathbb{R}^p} \left\{\cos\left(\omega^\top x\right) \cos\left(\omega^\top x'\right) + \sin\left(\omega^\top x\right) \sin\left(\omega^\top x'\right)\right\} d\Lambda(\omega) \quad (9.19) \end{aligned}$$

for some positive measure (w.l.o.g. a probability distribution) Λ called *spectral measure* of k. For many widely used kernels, spectral measure takes a simple form, e.g. if $k(x, x') = \exp\left(-\frac{1}{2\gamma^2} \|x - x'\|_2^2\right)$, then Λ is a multivariate normal $\mathcal{N}(0, \gamma^{-2} I)$. Now, for a given Λ, we sample m frequencies $\{\omega_j\} \sim \Lambda$ and use a Monte Carlo estimator of the kernel function given by the integral in (9.19):

$$\begin{aligned} \tilde{k}(x, y) &= \frac{1}{m} \sum_{j=1}^{m} \left\{\cos\left(\omega_j^\top x\right) \cos\left(\omega_j^\top y\right) + \sin\left(\omega_j^\top x\right) \sin\left(\omega_j^\top y\right)\right\} \\ &= \langle \phi_\omega(x), \phi_\omega(y) \rangle_{\mathbb{R}^{2m}}, \end{aligned}$$

which is an approximate kernel corresponding to an explicit set of features $\phi_\omega(x) \in \mathbb{R}^{2m}$ given by

$$x \mapsto \frac{1}{\sqrt{m}} \left[\cos\left(\omega_1^\top x\right), \sin\left(\omega_1^\top x\right), \ldots, \cos\left(\omega_m^\top x\right), \sin\left(\omega_m^\top x\right)\right]$$

With this set of features, we can now run algorithms in the primal representation which is less costly than paying the computational cost in n.

10 Bayesian Optimization

10.1 Tuning hyperparameters as optimizing "black-box" functions

We have considered several families of machine learning models which have complex inference algorithms and often require tuning of a number of hyperparameters in order to make them work in practice. These could for example be kernel parameters, the number of layers and units per layer in a deep neural network, learning rates, regularization parameters, or batch sizes in stochastic optimizers. Many important implementation details of hyperparameter tuning are often missing and can be extremely time-consuming as the algorithm needs to be run repeatedly for a large number of hyperparameter configurations – one often uses some form of grid search or random search based on a particular objective function, such as cross-validated empirical risk. Can this be done in a more principled and automated way, i.e. without having "human in the loop"? Ideally, we would want a fully automated machine learning pipeline where (nearly) optimal model configuration is selected with a small number of algorithm runs.

More broadly, we are interested in optimizing a particular 'well behaved' (i.e. it exhibits some degree of smoothness but is nonconvex and possibly multimodal) function $f : \mathcal{X} \to \mathbb{R}$ over some bounded domain $\mathcal{X} \subset \mathbb{R}^d$, i.e. in solving[1]

$$x_\star = \text{argmin}_{x \in \mathcal{X}} f(x).$$

However, f is not known explicitly, i.e. it is a *black-box* function. We can only observe its potentially noisy pointwise evaluations $y_i = f(x_i) + \epsilon_i$ at selected locations x_i. Moreover, these pointwise evaluations may be extremely expensive (i.e. they correspond to training of a large machine learning model or even running a complex physical experiment - see [7] for further details).

10.2 Surrogate Gaussian Process models

The principle of Bayesian optimization is to use a surrogate probabilistic model of the black-box function f in order to carry out the optimization. The default choice for a surrogate model is a Gaussian process (GP) – being a flexible prior over functions (but other models are possible, based on random forests or student-t processes). Further, based on the GP model, we need to define a criterion, which we call an *acquisition*

[1] We will phrase the optimization, w.l.o.g. as minimization here. For maximization of f, we can just use minimization of $-f$.

function, which determines the next location in $\mathcal{X}$ where f will be evaluated. These locations will in certain sense be *most informative about the optimum of* f.

Good acquisition functions will need to balance exploration (learning more about f based on new evaluations) vs exploitation (finding the maximum based on the current model of f). Exploration-exploitation tradeoff will be based on our estimates of the uncertainty in the values of f - which is why GP regression models with closed-form uncertainty estimates are most commonly used in this context. Namely, it is customary to assume that the noise ϵ_i in the evaluations of the black-box function is i.i.d. $\mathcal{N}\left(0, \delta^2\right)$, to bring us to the vanilla GP regression context. We have seen in the previous chapter that the GP model

$$\begin{aligned} \mathbf{f} &\sim \mathcal{N}(0, \mathbf{K}) \\ \mathbf{y}|\mathbf{f} &\sim \mathcal{N}(\mathbf{f}, \delta^2 I), \end{aligned}$$

gives us a closed form expression for the posterior predictive mean $\mu(x)$ and the posterior predictive marginal standard deviation $\sigma(x) = \sqrt{\kappa(x,x)}$ at any new location x, i.e.

$$f(x)|\mathcal{D} \sim \mathcal{N}(\mu(x), \kappa(x,x)),$$

where

$$\begin{aligned} \mu(x) &= \mathbf{k}_{x\mathbf{x}}(\mathbf{K} + \delta^2 I)^{-1}\mathbf{y}, \\ \kappa(x,x) &= k(x,x) - \mathbf{k}_{x\mathbf{x}}(\mathbf{K} + \delta^2 I)^{-1}\mathbf{k}_{\mathbf{x}x} \end{aligned}$$

Exploration in this context means that we are seeking locations with high posterior variance $\kappa(x,x)$, exploitation that we are seeking location with low posterior mean $\mu(x)$.

10.3 Acquisition Functions

Most commonly used acquisition functions are: *GP-LCB*, *probability of improvement (PI)* and *expected improvement (EI)*.

GP-LCB

GP-LCB follows the principle of the "optimism in the phase of uncertainty" and simply seeks to minimize the lower $(1-\alpha)$-credible bound of the posterior of the unknown function values $f(x)$, i.e.

$$\alpha_{LCB}(x) = \mu(x) - z_{1-\alpha}\sigma(x),$$

where $z_{1-\alpha} = \Phi^{-1}(1-\alpha)$ is the desired quantile of the standard normal distibution. Note that the same principle is used extensively in the theory of K-armed bandits which deals with a similar setup of the optimization of uncertain objectives over a discrete number of choices (not covered in this course), but is typically referred to as UCB (upper confidence/credible bound) since the convention is to consider maximization rather than minimization.

Probability of improvement (PI)

If we denote by $\tilde{x}$ the optimal location so far, i.e. the observed $\tilde{y}$ is the minimum among $(y_1, \ldots, y_n)$. Consider $u(x) = \mathbf{1}\{f(x) < \tilde{y}\}$, i.e. u represents the indicator of the event that the function value at any given location is below the observed minimum. Then the probability of improvement is simply

$$\alpha_{PI}(x) = \mathbb{E}[u(x)|\mathcal{D}] = \int_{-\infty}^{\tilde{y}} \mathcal{N}\left(f; \mu(x), \sigma^2(x)\right) df = \Phi\left(\frac{\tilde{y} - \mu(x)}{\sigma(x)}\right).$$

Expected Improvement (EI)

Note that in the probability of improvement, we do not take into account the size of an improvement in the objective function. Thus, the PI method can be viewed as preferring exploitation - locations that have a high probability of being infinitesimally smaller than $\tilde{y}$ will be drawn over points that offer possibly larger improvement but less certainty. We wish to now use as the utility the expected improvement at the location x, i.e. we define

$$u(x) = \max(0, \tilde{y} - f(x)),$$

which measures the size of an improvement from the current (estimated) minimum $\tilde{y}$ and we seek to maximize $\mathbb{E}[u(x)|\mathcal{D}]$.

We will need the following simple result about a truncated normal random variable.

Lemma 24. *Let $S \sim \mathcal{N}(m, \tau^2)$. Denote by ϕ the density and by Φ the cdf of a standard normal random variable. Then*

$$\mathbb{E}[\max(0, S)] = \int_0^{\infty} s\mathcal{N}\left(s; m, \tau^2\right) ds = \Phi\left(\frac{m}{\tau}\right) m + \phi\left(\frac{m}{\tau}\right) \tau.$$

As a shorthand, we will write $\gamma(x) = \frac{\tilde{y} - \mu(x)}{\sigma(x)}$.
Now,

$$\begin{aligned} \alpha_{EI}(x) = \mathbb{E}[u(x)|\mathcal{D}] &= \int_{-\infty}^{\tilde{y}} (\tilde{y} - f)\mathcal{N}\left(f; \mu(x), \sigma^2(x)\right) df \\ &= \int_0^{\infty} s\mathcal{N}\left(s; \tilde{y} - \mu(x), \sigma^2(x)\right) ds \end{aligned}$$

after the substitution $s = \tilde{y} - f$. Now, by Lemma 24 we get the most commonly used expression for the expected improvement acquisition function:

$$\begin{aligned} \alpha_{EI}(x) &= \Phi(\gamma(x))(\tilde{y} - \mu(x)) + \phi(\gamma(x))\sigma(x) \\ &= \sigma(x)(\gamma(x)\Phi(\gamma(x)) + \phi(\gamma(x))). \end{aligned}$$

While it is the most commonly used, the above expression is ignoring the noise in $f(\tilde{x})$, i.e. it is treating $\tilde{y}$ as the actual value of the objective which can be problematic if the noise level in evaluations is not negligible. Naively, we could simply replace $\tilde{y}$ with the predictive mean $\mu(\tilde{x})$. However, the predictive variance is still ignored.

Exercise 25. Derive the alternative expression for both the probability of improvement and for the expected improvement acquisition functions by considering the distribution of $u(x) = \mathbf{1}\{f(x) < f(\tilde{x})\}$. In particular, show that

$$\mathbb{E}[u(x)|\mathcal{D}] = \Phi(\gamma(x))(\mu(\tilde{x}) - \mu(x)) + \phi(\gamma(x))\rho(x, \tilde{x}), \tag{10.1}$$

where $\rho(x, \tilde{x}) = \sqrt{\kappa(x, x) + \kappa(\tilde{x}, \tilde{x}) - 2\kappa(x, \tilde{x})}$.

Note that we can interpret the two terms in 10.1 as trading-off exploration vs. exploitation. We can increase the acquisition either by decreasing $\mu(x)$ (exploitation) or by increasing $\rho(x, \tilde{x})$, i.e. a notion of a distance from the current optimum $\tilde{x}$ (exploration).

Bibliography

[1] David Arthur and Sergei Vassilvitskii. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, 2007.

[2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[3] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, December 2006.

[4] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer, 2004.

[5] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[6] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[7] Eric Brochu, Vlad M Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. eprint arXiv:1012.2599, arXiv.org, December 2010.

[8] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *The 4th International Conference on Learning Representations (ICLR)*, 2016.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[10] Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via Maximum Mean Discrepancy optimization. In *Proc. Uncertainty in Artificial Intelligence (UAI)*, 2015.

[11] Fajwel Fogel, Alexandre d'Aspremont, and Milan Vojnovic. Spectral ranking using seriation. *Journal of Machine Learning Research*, 17(88):1–45, 2016.

[12] Arthur Gretton. Lecture notes on Reproducing kernel Hilbert spaces in Machine Learning, University College London, 2017. `http://www.gatsby.ucl.ac.uk/~gretton/coursefiles/rkhscourse.html`.

[13] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

[14] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *The 2nd International Conference on Learning Representations (ICLR)*, 2014.

[15] Jon M. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems 15*, pages 463–470. MIT Press, 2003.

[16] Brian Kulis and Michael I. Jordan. Revisiting k-means: New Algorithms via Bayesian Nonparametrics. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 513–520, 2012.

[17] Prem Melville and Vikas Sindhwani. Recommender systems. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 829–838. Springer US, Boston, MA, 2010.

[18] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[19] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007.

[20] C.E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[21] Danilo Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[22] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural Comput.*, 11(2):305–345, February 1999.

[23] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

[24] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10:1299–1319, 1998.

[25] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[26] M. E. Tipping and Christopher Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 21/3:611–622, January 1999.

[27] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.