SC4/SM8 Advanced Topics in Statistical Machine Learning

Department of Statistics, University of Oxford
`https://github.com/ywteh/advml2020`

Lecturer: Yee Whye Teh

Hilary Term 2020

Contents

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 (iid)* 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 a simple algorithm for each: *Principal Components Analysis (PCA)* for dimensionality reduction and the *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

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

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) $v_1 \in \mathbb{R}^p$ is (a unit length vector in) 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 *score matrix*

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

Deriving the first principal component

Recall that we model our dataset as an iid 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 a random vector $Z = V_{1:k}^\top X$ which is a k-dimensional random vector. Dimensions of Z are called **derived variables**. Consider the first dimension of Z:

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

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:

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

and the corresponding vector of partial derivatives

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

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.

Similarly, we can show that each subsequent PC is given by the eigenvector corresponding to the next largest eigenvalue (problem sheet shows this for the second PC). As a result, we obtain the *eigenvalue decomposition* of S given by

$$S = V \Lambda V^\top \tag{1.8}$$

where Λ is a diagonal matrix with eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0 \tag{1.9}$$

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

$$\hat{x}_i = V_{1:k} V_{1:k}^\top x_i. \tag{1.10}$$

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

$$\mathbf{Z} = \mathbf{X}V = UDV^\top V = UD, \tag{1.11}$$

If $p \leq n$ this means

$$z_i = [U_{i1}D_{11}, \ldots, U_{ip}D_{pp}]^\top, \tag{1.12}$$

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

$$z_i = [U_{i1}D_{11}, \ldots, U_{in}D_{nn}, 0, \ldots, 0]^\top. \tag{1.13}$$

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.

Lack of Axiomatic Definition

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 $F : (\mathcal{D} = \{x_i\}_{i=1}^{n}, \rho) \mapsto \{C_1, \ldots, C_K\}$ which takes as an input dataset $\mathcal{D}$ and a dissimilarity function ρ and returns a partition of $\mathcal{D}$:

- **Scale invariance.** For any $\alpha > 0$, $F(\mathcal{D}, \alpha\rho) = F(\mathcal{D}, \rho)$, 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 $F(\mathcal{D}, \rho) = C$, 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:

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

  then $F(\mathcal{D}, \rho') = F(\mathcal{D}, \rho)$. 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, and there are algorithms satisfying any two of them, 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 two extensions.

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 C_k 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:

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

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

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

Thus one can employ an iterative alternating optimization, which is exactly the K-means algorithm given in Algorithm 1.1.

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.

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 $\{C_1, \ldots, C_K\}$
```

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

$$\rho_i^2 = \min\left\{\|x_i - \mu_1\|_2^2, \ldots, \|x_i - \mu_{k-1}\|_2^2\right\}. \tag{1.14}$$

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

$$\mathbb{E}\left[W\left(\{C_k^{++}\}\right)\right] \leq 8\left(\log K + 2\right) W^*, \tag{1.15}$$

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 (see Chapter **??**). 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 $i = 1, \dots, n$,
   - if $\min_{k=1,\dots,K} \|x_i - \mu_k\|_2^2 > \lambda$, set $K \leftarrow K + 1$, $c_i \leftarrow K$, $\mu_K \leftarrow x_i$
   - otherwise, set $c_i = \operatorname{argmin}_{k=1,\dots,K} \|x_i - \mu_k\|_2^2$, and update the previous and current cluster centroids that item i belongs to.
3. Repeat Step 2 until it stops changing the cluster assignments for all items.

The tuning parameter λ controls the tradeoff between the traditional K-means objective and the number of clusters, and can be interpreted as a cost associated with each cluster used. Like K-means, DP-means can be shown to terminate after a finite number of iterations (problem sheet).

1.2 Supervised Learning Basics

1.2.1 Empirical Risk Minimisation (ERM)

Loss and Risk

In **supervised learning**, 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. It is typically 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 $\mathcal{X} \times \mathcal{Y}$. The goal of supervised learning is to find the function f which *minimizes the expectation of the loss* over $P_{X,Y}$, which is called *risk*. In machine learning, if the output space is discrete, this is called **classification**, while **regression** refers to the case the output space is continuous.

Empirical Risk Minimisation (ERM)

A **loss function** is any function

$$L : \mathcal{Y} \times \mathcal{Y} \times \mathcal{X} \to \mathbb{R}^+. \tag{1.16}$$

The loss $L(y_i, f(x_i), x_i)$ measure the discrepancy between predicted output values $f(x_i)$ and the true output values y_i at the inputs x_i.

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

$$R(f) = E_{X,Y} L(Y, f(X), X). \tag{1.17}$$

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

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i), x_i). \tag{1.18}$$

The **empirical risk minimisation** (ERM) problem aims to find the function with minimum risk:

$$\hat{f} = \operatorname{argmin}_{f \in \mathcal{H}} \hat{R}(f), \tag{1.19}$$

where $\mathcal{H}$ is a given class of functions (called the **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 functions 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 class 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; see Chapter 3.

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 $yf(x)$.

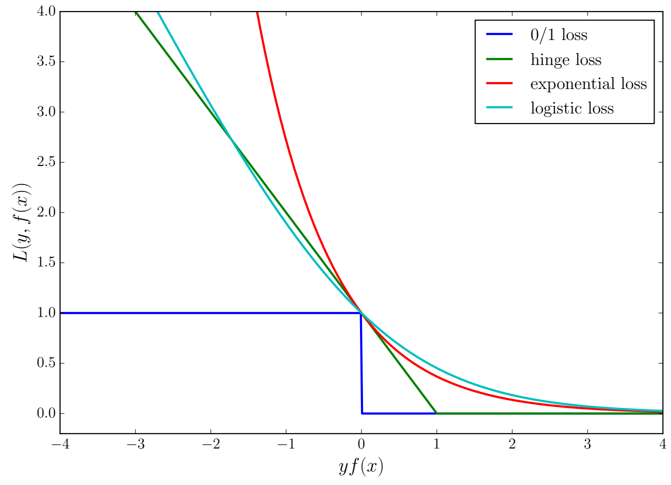

Figure 1.1: Loss functions for binary classification

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**. The optimal solution is called the **Bayes classifier** and is given by $f(x) = \operatorname{argmax}_{k\in\{0,1\}} \mathbb{P}(Y = k|X = x)$),
  - **hinge loss**: $L(y, f(x)) = (1 - yf(x))_+$
    (used in *support vector machines* - leads to sparse solutions),
  - **exponential loss**: $L(y, f(x)) = e^{-yf(x)}$
    (used in *boosting* algorithms, e.g. 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 $\text{med}[Y|X = x]$),

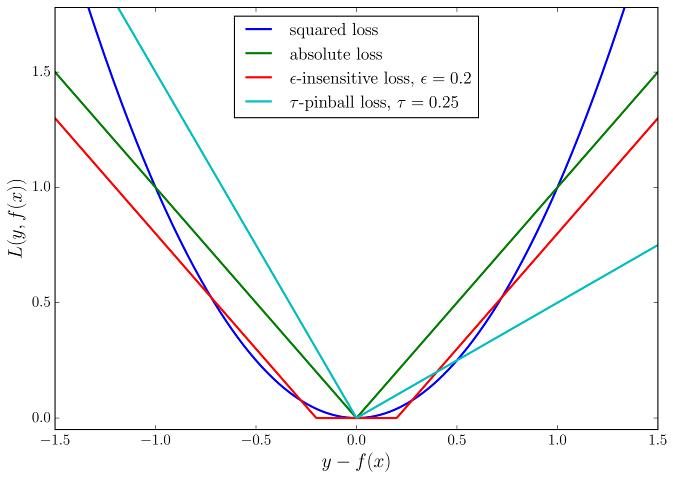

Figure 1.2: Loss functions for regression

- 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 loss** or **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, logistic[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

$$\frac{\partial^2 \hat{R}}{\partial w \partial w^\top} = \frac{1}{n}\sum_{i=1}^{n} \rho''\left(y_i w^\top \varphi(x_i)\right) \varphi(x_i)\varphi(x_i)^\top, \tag{1.20}$$

[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.

[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

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 other examples drawn from $P_{X,Y}$.

The idea behind **regularisation** is to limit the flexibility of the hypothesis class in order to prevent overfitting. For example, for the hypothesis space $\mathcal{H} = \{f_\theta : \theta \in \mathbb{R}^p\}$ consisting of functions parameterised by a vector θ, this can be achieved by adding a term which *penalises large values for the 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.

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**

$$\|f\|_{W^1}^2 = \int_{-\infty}^{+\infty} f(x)^2 dx + \int_{-\infty}^{+\infty} f'(x)^2 dx, \tag{1.21}$$

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

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top x_i - b)^2 + \frac{\lambda}{n} \|w\|_2^2. \tag{1.22}$$

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

$$\min_{w} \|\mathbf{y} - \mathbf{X}w\|_2^2 + \lambda \|w\|_2^2. \tag{1.23}$$

Differentiating and setting to zero gives the closed form solution

$$w = \left(\mathbf{X}^\top \mathbf{X} + \lambda I\right)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{1.24}$$

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 $f(x) = w^\top x + b$,

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

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:

$$\log \frac{p(y_i = +1 | w, b, x_i)}{p(y_i = -1 | w, b, x_i)} = w^\top x_i + b. \tag{1.26}$$

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

$$p(y_i = +1 | w, b, x_i) = \frac{1}{1 + e^{-(w^\top x_i + b)}} = \sigma(w^\top x_i + b), \tag{1.27}$$

$$p(y_i = -1 | w, b, x_i) = \frac{1}{1 + e^{w^\top x_i + b}} = \sigma(-w^\top x_i - b), \tag{1.28}$$

where we denoted by $\sigma(t) = 1/(1 + e^{-t})$ the **logistic function** (aka **sigmoid function**) which maps the real line to the $(0, 1)$ interval. Note that the logistic function satisfies

$\sigma(-t) = 1-\sigma(t)$. Thus, we can write (1.27) and (1.28) 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

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i(w^\top x_i + b)\} + \frac{\lambda}{n}\|w\|_2^2. \tag{1.29}$$

This does not have a closed form solution and requires numerical optimisation.

Eq. (1.29) is not how you would typically see an SVM written in the literature, though. In the next chapter, we will make a deep dive into SVMs, introducing it from the perspective 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:

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

We denote $\mathcal{D} := \bigcap_{i=0}^{m} \mathrm{dom} f_i \cap \bigcap_{j=1}^{r} \mathrm{dom} h_j$, and require the domain $\mathcal{D} \subseteq \mathbb{R}^n$ where the objective function f_0 and the constraint functions $f_1, \dots, f_m, h_1, \dots, h_r$ 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 $(\lambda, \nu) \in \mathrm{dom} g$.

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

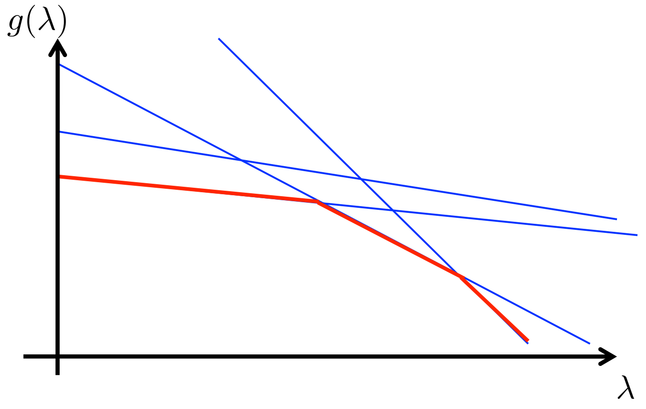

Figure 2.1: Example: Lagrangian with one inequality constraint, $L(x, \lambda) = f_0(x) + \lambda f_1(x)$, where x here can take one of four values for ease of illustration. The infimum of the resulting set of four affine functions is concave in λ.

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. □

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

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

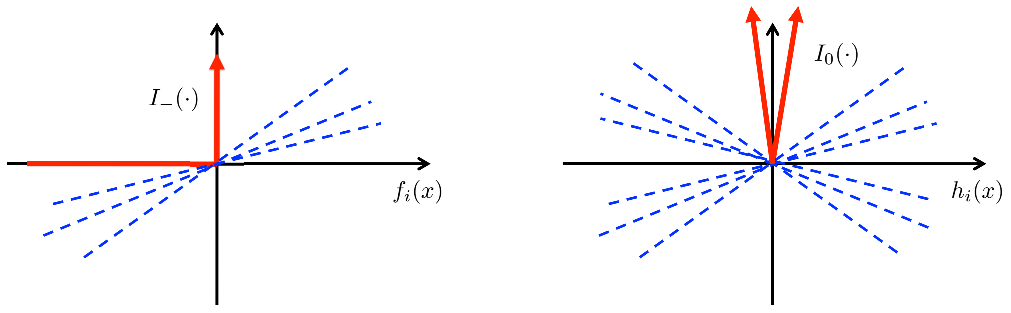

Figure 2.2: Linear lower bounds on indicator functions. Blue functions represent linear lower bounds for different slopes λ and ν, for the inequality and equality constraints, respectively.

where

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

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 **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:

$$d^* \le p^*.$$

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

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

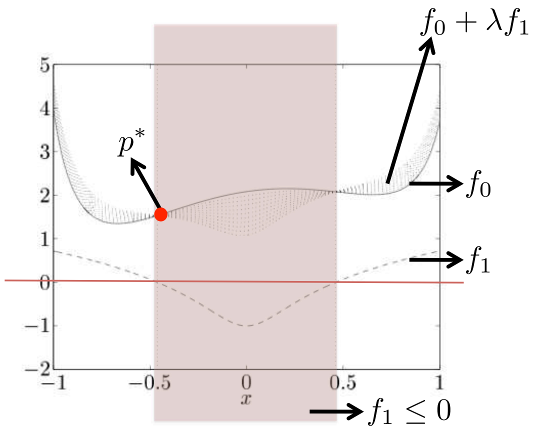

Figure 2.3: Illustration of the Lagrangian on a simple problem with one inequality constraint (from [6, Fig. 5.1]).

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

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

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 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

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

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**:

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

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

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

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

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

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 the 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 Figure 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**, which is defined as twice the *smallest* distance from each class to the separating hyperplane (see Figure 2.4).

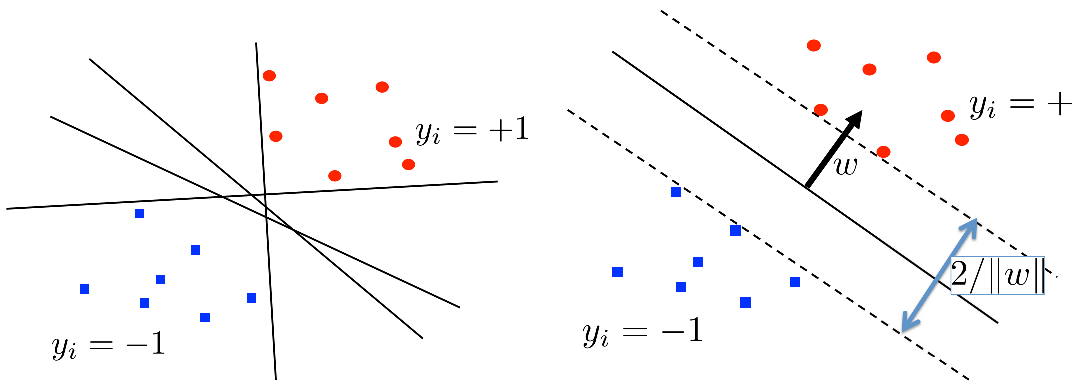

Figure 2.4: The linearly separable case. There are many linear separating hyperplanes, but only one maximum margin separating hyperplane.

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

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

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.

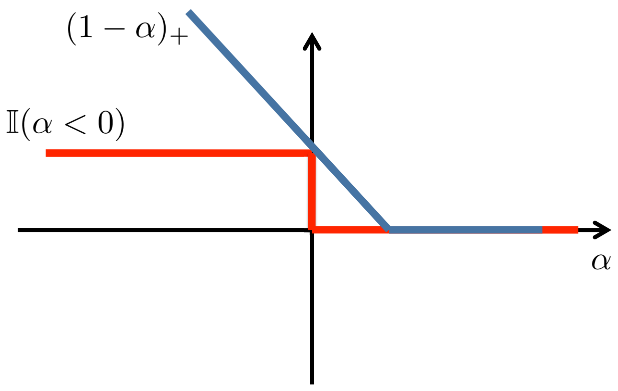

Figure 2.5: The hinge loss is an upper bound on the 0/1 loss.

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.

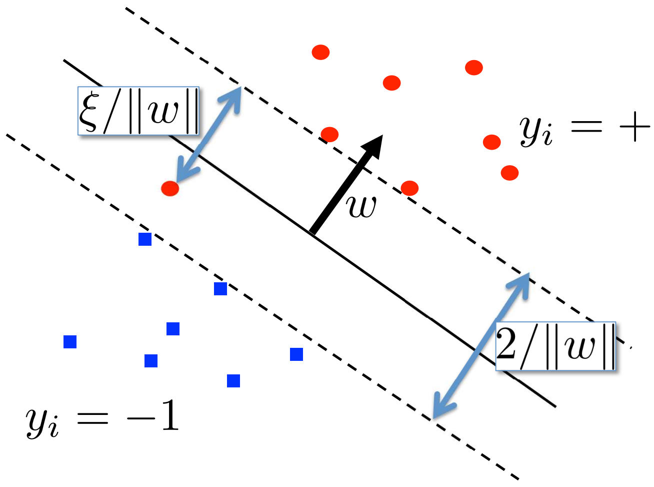

Figure 2.6: The nonseparable case. Note the red point which is a distance $\xi/\|w\|$ from the margin.

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. This formulation is known as the C-SVM.

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

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

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 (problem sheet):

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

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 non-linear 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. 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 $\mathcal{H}$ and a map $\varphi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$,

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

We will call the Hilbert space associated with 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 vector-valued 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. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is **positive definite** if $\forall n \geq 1$, $\forall (a_1, \ldots a_n) \in \mathbb{R}^n$, $\forall (x_1, \ldots, x_n) \in \mathcal{X}^n$,

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

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. Let $\mathcal{H}$ be a *Hilbert space of functions* $f : \mathcal{X} \to \mathbb{R}$ 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*).

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

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

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, the **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

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

we have

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

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

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

Then

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

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 kernel**, **RBF kernel**, **squared exponential kernel** or **exponentiated quadratic kernel**, 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 kernel**, 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

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

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 is the **rational quadratic kernel** 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]):

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

for the RKHS $\mathcal{H}$ with kernel $k(x,x')$. This **kernel 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 $\|w\|^2_{\mathcal{H}}$: 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 $\mathbf{S}v_m = \lambda_m v_m$, $m = 1,\dots,M$, and we need the top $k \ll \min\{n, M\}$ eigenvectors v_m, $m = 1,\dots,k$, 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 $\operatorname{span}\{\varphi(x_i) : i = 1,\dots,n\}$, 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 $j = 1,\dots,n$. This gives

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

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*

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

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

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

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

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

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):*

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

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**):

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

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.

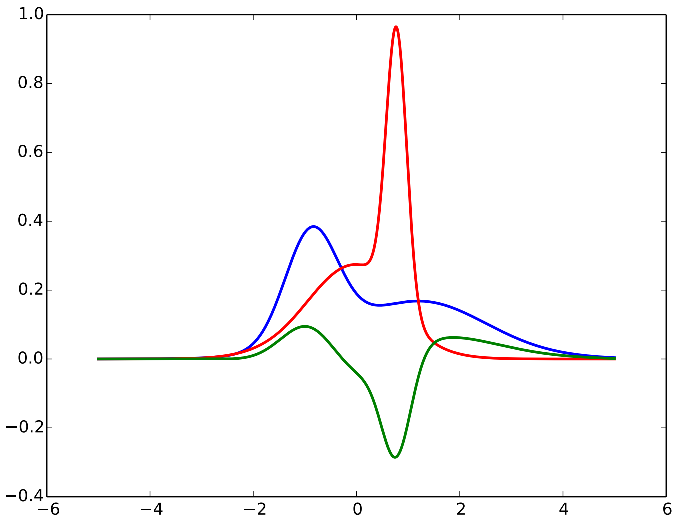

Figure 3.1: Witness function for a difference between two univariate densities

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 $MMD_k\left(P,Q\right)=0$ 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

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

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

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

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:

$$\widehat{\text{first term}} = \frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m k_{ij} l_{ij} = \frac{1}{m^2} \operatorname{tr}(KL)$$

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

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

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

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

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

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, $\mathrm{tr}\left(\left(HKH\right)\left(HLH\right)\right) = \mathrm{tr}\left(H\left(KHLH\right)\right) = \mathrm{tr}\left(KHLHH\right) = \mathrm{tr}\left(KHLH\right)$.

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:

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

which has an intuitive explanation of how it measures the dependence between X and Y. Namely, the function $(A,B) \to \mathrm{tr}\left(A^T B\right)$ 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.

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.

Index