THE BHATTACHARYYA KERNEL BETWEEN SETS OF VECTORS

YJ (YO JOONG) CHOE

Abstract. In machine learning, we wish to find general patterns from a given set of examples, and in many cases we can naturally represent each example as a set of vectors rather than just a vector. In this expository paper, we show how to represent a learning problem in this "bag of tuples" approach and define the Bhattacharyya kernel, a similarity measure between such sets related to the Bhattacharyya distance. The kernel is particularly useful because it is invariant under small transformations between examples. We also show how to extend this method using kernel trick and still compute the kernel in closed form.

Contents

1. Introduction

In machine learning and statistics, we utilize *kernels*, which are measures of "similarity" between two examples. Kernels take place in various learning algorithms, such as the perceptron and support vector machines. In the standard case where we represent each example as a point in $\mathbb{R}^n$, we use kernels such as the dot product $K(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$ and the Gaussian radial basis function (RBF) kernel $K(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x}-\mathbf{x}'\|^2/2\sigma^2}$ (σ is a constant). Using different kernels, of course, leads to different outputs and some turn out to be more useful than others depending on types and patterns of the inputs.

In this paper, we present an approach to a learning problem that differs from the standard point-in-$\mathbb{R}^n$ approach and consequently a different kernel that suits this approach. We represent each example not as a point in $\mathbb{R}^n$ but as a *"bag of tuples"*, or a set of vectors in $\mathbb{R}^n$. The purpose is to create a method to find a similarity measure with "soft invariance", e.g. invariance between translated, rotated, or warped images.

The standard kernels do not work because each example is no longer a vector. To deal with this problem and to achieve the soft invariance property, we fit a distribution to each set of vectors and define a kernel between distributions. Here, our choice is the *Bhattacharyya distance*, which is a concept in statistics that measures similarity between two distributions over the same space. Consequently, we call this kernel the *Bhattacharyya kernel*.

Finally, we explain how to incorporate the standard *kernel trick* in this process - in particular, we show how a base kernel and a corresponding feature map may take place in the process of fitting a distribution to a set of vectors in $\mathbb{R}^n$. In all these cases, we are able to compute the kernel between distributions in closed form.

2. Background

2.1. **Learning.** The problem of learning involves finding patterns about an object or an event given its instances. This inductive process is one of everyday human practices: for example, one recognizes an animal walking by as a dog, because (according to what he observed before) animals with such looks and behaviors tend to be dogs rather than any other animals.

But now we are looking for a systematic algorithm that can find such patterns, less magically than how our brain does it. Of course, we still want the patterns to be correct most of the time, or at least as much as our brain does. This very general problem can arise in almost any setting, and it is conveniently described and solved in terms of mathematics.

There are different kinds of learning problems depending on how one categorizes them, but the most natural one can be described as the following: *given inputs $x_1, x_2, \ldots, x_m$ and the corresponding outputs $y_1, y_2, \ldots, y_m$, find a function $h : x \mapsto y$ that most accurately finds the correct y for* ***any*** *given x.*

From the previous example, each observation of some animal (its looks, behaviors, or any other features) becomes each x_i, and whether or not it was actually a dog becomes the corresponding y_i. (Thus, at least in this case, we are only considering those observations for which one could later figure out that the animal was or was not actually a dog.)

We will formally define this type of a learning problem and the relevant terminologies:

Definition 2.1 (supervised learning problem)**.** A **supervised learning problem** in a *statistical* learning model is any problem that asks to find

$$\begin{aligned}\hat{h} &= \underset{h:x\mapsto y}{\operatorname{argmin}}\ \mathcal{E}_{\text{true}}(h)\\ &= \underset{h:x\mapsto y}{\operatorname{argmin}}\ \underset{(x,y)\sim p}{\mathbb{E}}\ \ell(h(x),y)\end{aligned}$$

given a **loss function** $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ and a set of **examples** sampled independently from some unknown distribution p over $\mathcal{X} \times \mathcal{Y}$:

$$(x_1, y_1), \ldots, (x_m, y_m) \overset{\text{IID}}{\sim} p.$$

Here, $\mathcal{X}$ is called the **input space** and its elements are called **inputs**. Similarly, $\mathcal{Y}$ is called the **output space** and its elements are called **outputs** or **labels**. The set of given examples $S_{train} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ is denoted as the **training set**.

Any function h that maps x to y is called the **hypothesis**. The space of functions $\mathcal{H}$ from which the solver decides to find $\hat{h}$ is called the **hypothesis space**. For a hypothesis h, $\mathcal{E}_{\text{true}}(h) = \mathbb{E}_{(x,y)\sim p}\,\ell(h(x), y)$ is called the **true error** of h.

The standard input space is $\mathcal{X} = \mathbb{R}^n$. *In fact, we can represent almost all inputs of a supervised learning problem in* $\mathbb{R}^n$. In the dog's case, we can represent each observed animal as a point in $\mathbb{R}^n$ by including some factors about that animal as coordinates and labeling each within a subset of the real numbers. For example, we may represent each observation to be (number of legs, color of furs - 1 if white, 2 if brown, etc., height), and we could have $(4, 1, 5)$, $(4, 2, 10)$, and so on.

A typical output space is $\mathcal{Y} = \{1, -1\}$, the simplest case in which we call the problem a **(binary) classification problem**. In our example, for each observation we may label 1 if the animal is a dog and -1 otherwise. Other standard output spaces are $\mathcal{Y} = \{1, 2, \ldots, k\}$ (multi-class classification) and $\mathcal{Y} = \mathbb{R}$ (regression). For the purpose of this paper, we will stick with the simplest one, although we do note here that the techniques can readily extend to the other two.

There are various kinds of loss functions, such as the zero-one loss:

$$\ell_{0/1}(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y,\\ 1 & \text{if } \hat{y} \neq y.\end{cases}$$

But with any loss function, the true error for each hypothesis h cannot be computed because p is unknown. In practice, we estimate the true error based on some of the given examples and put bounds on this estimate.

2.2. **Examples of learning algorithms.** Now that we discussed what a learning problem is, we will introduce two algorithms that actually solve learning problems. In both cases, we will assume that the input space is $\mathbb{R}^n$ and the output space is $\{1, -1\}$.

The **perceptron** is perhaps the classic learning algorithm that classifies points in $\mathbb{R}^n$ each labeled either 1 or -1. In its simplest form, the perceptron tries to find a linear hyperplane in $\mathbb{R}^n$ that separates points labeled 1 from points labeled -1. Most importantly, the perceptron learns from mistakes: at every mistake, it adjusts its hyperplane so that it correctly classifies at least the ones that it had seen before.

Note that this algorithm does not always work for any set of labeled points in $\mathbb{R}^n$: it requires that the points are separable by some linear hyperplane in the first place.

Example 2.2 (the perceptron, Rosenblatt, 1957).

```
GIVEN: $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ and $y_1, \ldots, y_m \in \{1, -1\}$.
$\mathbf{w} \leftarrow (0, 0, \ldots, 0)$
$t \leftarrow 1$
FOR each example $(\mathbf{x}_i, y_i)$:
    IF $(\mathbf{w} \cdot \mathbf{x}_i \geq 0)$ predict $\hat{y}_i = 1$; ELSE predict $\hat{y}_i = -1$
    IF $(\hat{y}_i = -1$ and $y_i = 1)$ let $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_i$
    IF $(\hat{y}_i = 1$ and $y_i = -1)$ let $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_i$
RETURN the hypothesis $h(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x})$
```

This algorithm works for points that are separable by a linear hyperplane passing through the origin. Here, $\mathbf{w}$ represents the normal vector to the linear hyperplane that the algorithm is constructing. The algorithm predicts $\hat{y}_i = 1$ if the point $\mathbf{x}_i$ is to the right of the current linear hyperplane $(\mathbf{w} \cdot \mathbf{x}_i \geq 0)$ and $\hat{y}_i = -1$ otherwise. Upon mistakes, if what it predicted as 1 is actually labeled -1, then the algorithm adjusts its hyperplane by simply adding the point vector to the normal vector, and it similarly adjusts by subtracting in the other case. It is known that this algorithm works surprisingly well at least for examples that are linearly separable, especially with large margins.

What we are interested in for the purpose of this paper, however, is not exactly why the algorithm works but how it utilizes a similarity measure: the dot product. First notice that each prediction (classification) is based on whether or not the dot product between $\mathbf{w}$ and $\mathbf{x}_i$ is positive, i.e. how similar the point is to the normal vector. But more importantly, the normal vector is just a linear combination of the previous examples $\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$, so that $\mathbf{w} \cdot \mathbf{x}_i$ is actually $\sum_{j=1}^{i-1} \alpha_j (\mathbf{x}_j \cdot \mathbf{x}_i)$, where α_j is either 1, -1, or 0 depending on whether the point $\mathbf{x}_j$ was added to or subtracted from $\mathbf{w}$, or was not used at all. This gives a very important observation: *the classification process depends only on the similarity (in dot product) between inputs $x_1, \ldots, x_m$, not on the value of each x_i itself.*

We can also notice a similar phenomenon in a more sophisticated and useful algorithm called the **support vector machine (SVM)**. One can view this algorithm as a much improved version of the perceptron, because it also finds a linear classifier (hyperplane) but more elegantly. Its goal is to actually maximize the "distance" between the hyperplane and the point closest to it. This distance is called the margin, and the point closest to the optimal hyperplane is called the support vector. This problem can be stated as the following optimization problem:

$$\underset{\mathbf{w} \in \mathbb{R}^n,\ b \in \mathbb{R}}{\text{maximize}}\ \delta \quad \text{s.t.} \quad \frac{y_i(\mathbf{w} \cdot \mathbf{x}_i + b)}{\|\mathbf{w}\|} \geq \delta \quad \forall i = 1, 2, \ldots, m.$$

Notice that $y_i(\mathbf{w} \cdot \mathbf{x}_i + b)/\|\mathbf{w}\|$ measures the distance between the point $\mathbf{x}_i$ and the linear hyperplane with normal vector $\mathbf{w}$ and constant term b (the y_i term requires the hyperplane to actually correctly classify all the points in the first place - the fraction is nonnegative for all i only if all the classification is correct).

Again, we will not get into the details (for detailed explanation, see [7]), but we do note that the SVM also depends heavily on the similarity in dot product between inputs. The (Lagrangian) dual problem that we actually solve in the SVM is given by

$$\underset{\alpha_1,\ldots,\alpha_m}{\text{maximize}}\ L(\alpha_1,\ldots,\alpha_m) = \sum_{i=1}^m y_i\alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j)$$

$$\text{s.t.}\quad \sum_{i=1}^m \alpha_i = 0 \quad \text{and} \quad y_i\alpha_i \geq 0 \quad \forall i = 1,2,\ldots,m.$$

Here again, the algorithm no longer requires the knowledge of the value of each $\mathbf{x}_i$, but only the dot product between inputs. The idea, therefore, is to exploit this fact and *generalize* this similarity measure.

2.3. **Kernels.** We formalize the similarity measure by the notion of what is called a kernel (different from the kernel in algebra). The idea is to simply replace the dot product with a kernel and tackle the problems in which linear classifiers were just not suitable.

Definition 2.3 (kernel)**.** A **kernel** is a function $k : \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ that satisfies the following properties:

(1) **symmetric**: for all $x, x' \in \mathcal{X}$, $k(x,x') = k(x',x)$;
(2) **positive semi-definite (PSD)**: for any $m \in \mathbb{N}$ and for all $x_1,\ldots,x_m \in \mathcal{X}$ and $c_1,\ldots,c_m \in \mathbb{R}$, $\sum_{i=1}^m\sum_{j=1}^m c_ic_jk(x_i,x_j) \geq 0$.

That is, a kernel is any symmetric and positive semi-definite (SPSD) function that maps any pair of inputs in an input space to a real number. Recall that a square matrix $K \in \mathbb{R}^{m\times m}$ is positive semi-definite if $\mathbf{z}^TK\mathbf{z} = \sum_{i=1}^m\sum_{j=1}^m z_iz_jK_{ij} \geq 0$ for all $\mathbf{z} = (z_1,\ldots,z_m)^T \in \mathbb{R}^m$. Thus, we see how this definition of a PSD function is an analogue of the concept for a matrix. Same holds for the concept of symmetry.

Example 2.4. These are a few standard kernels, where $\mathcal{X} = \mathbb{R}^n$:

(1) the **linear** kernel (i.e., the dot product):

$$k(\mathbf{x},\mathbf{x}') = \mathbf{x}\cdot\mathbf{x}'$$

(2) the **polynomial** kernel (degree d):

$$k(\mathbf{x},\mathbf{x}') = (1+\mathbf{x}\cdot\mathbf{x}')^d$$

(3) the **Gaussian** radial basis function (RBF) kernel (parameter σ):

$$k(\mathbf{x},\mathbf{x}') = e^{-\|\mathbf{x}-\mathbf{x}'\|^2/2\sigma^2}$$

where $\mathbf{x},\mathbf{x}' \in \mathbb{R}^n$ and $\|\cdot\|$ in (3) is the Euclidean norm. It is not difficult to show that all are SPSD.

Again, a kernel is a generalization of the dot product, and it serves as a similarity measure. This implies its relevance to the more familiar concept of an inner product. In fact, we may construct a Hilbert space from a SPSD kernel, just as we construct a Hilbert space from an inner product (positive definite Hermitian form). The key idea is simple: because the kernel is SPSD, we can easily show that it serves as an inner product in some Hilbert space $\mathcal{F}$.

Theorem 2.5 (feature space and feature map)**.** *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel. Then, there exists a Hilbert space $\mathcal{F}$, which we denote as a **feature space**, and a corresponding **feature map** $\varphi : \mathcal{X} \to \mathcal{F}$, such that*

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{F}},$$

where $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ denotes the inner product in $\mathcal{F}$.

Proof. There are many different ways to construct a feature map and a feature space given a kernel. It is not difficult to find one given any kernel, but here we will construct a more useful and particularly important one. The following proof shows an explicit construction of a feature space that is often called the **reproducing kernel Hilbert space (RKHS)** of kernel k.

For each $x \in \mathcal{X}$, define the function $k_x(\cdot) = k(x, \cdot)$. Then, let $\mathcal{F}_{\text{pre}}$ be the space of all finite linear combinations of such functions, i.e.

$$\mathcal{F}_{\text{pre}} = \{\sum_{i=1}^{n} \alpha_i k_{x_i} \mid x_1, \dots, x_n \in \mathcal{X} \text{ and } \alpha_1, \dots, \alpha_n \in \mathbb{R}\}.$$

Now, for all $x, x' \in \mathcal{X}$, define

$$\langle k_x, k_{x'} \rangle_{\mathcal{F}} = k(x, x'),$$

and for all $x_1, \dots, x_n \in \mathcal{X}, \alpha_1, \dots, \alpha_n \in \mathbb{R}$ and $x'_1, \dots, x'_m \in \mathcal{X}, \alpha'_1, \dots, \alpha'_m \in \mathbb{R}$, define

$$\left\langle \sum_{i=1}^{n} \alpha_i k_{x_i}, \sum_{j=1}^{m} \beta_j k_{x'_j} \right\rangle_{\mathcal{F}} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j \left\langle k_{x_i}, k_{x'_j} \right\rangle_{\mathcal{F}} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, x'_j).$$

Note that $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ is symmetric and positive definite by the property of our kernel k and is also linear by the above. That is, it is an inner product and we may complete $\mathcal{F}_{pre}$ with respect to $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ into a Hilbert space, which we will naturally denote as $\mathcal{F}$.

Finally, define the feature map $\varphi(x) = k_x$ for all $x \in \mathcal{X}$. □

Note that in a RKHS, for any $f = \sum_i \alpha_i k_{x_i} \in \mathcal{F}$ and $x \in \mathcal{X}$,

$$\langle f, k_x \rangle_{\mathcal{F}} = \left\langle \sum_i \alpha_i k_{x_i}, k_x \right\rangle_{\mathcal{F}} = \sum_i \alpha_i \langle k_{x_i}, k_x \rangle = \sum_i \alpha_i k(x_i, x) = \sum_i \alpha_i k_{x_i}(x) = f(x).$$

This crucial property tells that the kernel "represents" the evaluation of f at x.

But why do we need the idea of bringing a more complicated similarity measure when we have our familiar dot product already? This is because the use of feature maps to higher dimensions allows us to find more precise hypotheses by simply replacing the dot product with a more useful kernel. Figure 1 is a good example of a classification problem that is solved only by having a feature map to a higher-dimensional space.

What about all the complexity that an abstract Hilbert space may bring compared to the Euclidean space? We pointed out in the previous section that in most algorithms including the perceptron, *what we only need to compute is the inner product $\langle \varphi(x), \varphi(x') \rangle_{\mathcal{F}}$, i.e. the kernel value $k(x, x')$ - not the explicit value of each*

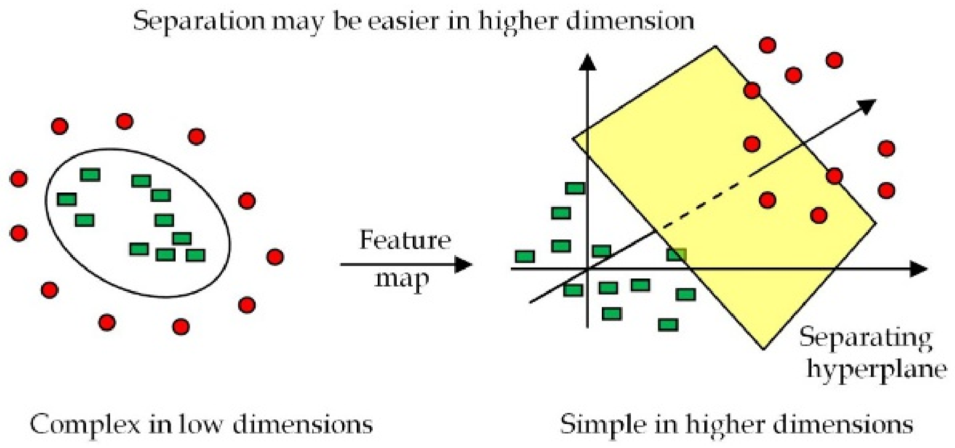
Separation may be easier in higher dimension
Feature map
Separating hyperplane
Complex in low dimensions
Simple in higher dimensions

FIGURE 1. A feature map from $\mathbb{R}^2$ to $\mathbb{R}^3$ that enables classification by a hyperplane in the feature space ($\mathbb{R}^3$). This can be done by simply replacing the dot product with a kernel that brings about a feature map to $\mathbb{R}^3$ by Theorem 2.4.

$\varphi(x)$ *in the Hilbert space* $\mathcal{F}$*.* This key observation allows us to utilize kernels involving high-dimensional or infinite-dimensional feature spaces in which all we care about is simply the inner product between two relevant elements and not the value of each element or the structure of the feature space. Of course, by Theorem 2.5, the inner product is precisely the value of the kernel. The whole process is thus called the **kernel trick**.

Now we go back to our discussion about learning in general. The reader may have noticed in Section 2.1 that the problem of learning is inherently similar to that of statistics: estimating parameters from known examples. This process appears in every approach to learning problems, and in the rest of Section 2 we review the basics of statistics that are relevant in this paper.

2.4. The normal distribution.

Many parts of learning algorithms do rely on techniques from statistics, and one very common and useful idea is, indeed, the normal (Gaussian) distribution.

Definition 2.6 (univariate normal distribution)**.** The **univariate normal distribution** with mean μ and standard variation σ is given by the probability density function

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}.$$

If $\mu = 0$ and $\sigma = 1$, this is often called the *standard normal distribution*. A random variable X is **normally distributed** with mean μ and standard variation σ (denoted as $X \sim \mathcal{N}(\mu, \sigma^2)$) if it obeys the above probability density function.

We generalize this concept to n-dimensions. A random vector is a column vector whose coordinates are each a random variable.

Definition 2.7 (covariance matrix)**.** The **covariance matrix** (or variance-covariance matrix) of a random vector $\mathbf{X} = (X_1, \ldots, X_n)^T$ is given by

$$\mathbf{\Sigma} = (\operatorname{Cov}(X_i, X_j))_{i,j=1}^n$$

where $\operatorname{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mathbb{E}X_i)(X_j - \mathbb{E}X_j)]$. In other words,

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

where $\mathbb{E}\mathbf{X} = (\mathbb{E}X_1, \ldots, \mathbb{E}X_n)^T$.

The latter definition implies that the covariance matrix is a generalization of variance ($\sigma^2 = \mathbb{E}[(X-\mu)^2]$) into n-dimensions. Note that any covariance matrix is SPSD: symmetry is obvious, and for $c = (c_1, \ldots, c_n)^T \in \mathbb{R}^n$,

$$c^T \mathbf{\Sigma} c = \sum_{i=1}^n \sum_{j=1}^n c_i c_j \operatorname{Cov}(X_i, X_j) = \operatorname{Var}(\sum_{i=1}^n c_i X_i) \geq 0.$$

Definition 2.8 (multivariate normal distribution)**.** A random vector $\mathbf{X} = (X_1, \ldots, X_n)^T$ has the **multivariate normal distribution** with mean $\boldsymbol{\mu} = (\mathbb{E}X_1, \ldots, \mathbb{E}X_n)^T$ and covariance matrix $\mathbf{\Sigma} = (\operatorname{Cov}(X_i, X_j))_{i,j=1}^n$ if every linear combination of $X_1, \ldots, X_n$ is normally distributed. This is denoted as $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$.

If the covariance matrix is non-degenerate (i.e., full rank), then the probability density function is given by

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} e^{-(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})/2}$$

where $\mathbf{x} \in \mathbb{R}^n$.

Note that, if $n = 1$, then this definition is consistent with that of the univariate normal.

2.5. **Maximum likelihood estimation (MLE).** Now we turn to perhaps the most standard technique in statistics: maximum likelihood estimation.

Definition 2.9 (parameters and parametric family)**.** The **parameters** of a distribution p are the indices that characterize the distribution within a certain family $\mathcal{P}$ of distributions. $\mathcal{P}$ is then called a **parametric family**. We denote the parameters of a distribution $p \in \mathcal{P}$ as a vector θ.

Example 2.10. Let $\mathcal{P}$ be the set of n-variate normal distributions. The parameters of each normal distribution $p \in \mathcal{P}$ are its mean $\boldsymbol{\mu}$ and its covariance matrix $\mathbf{\Sigma}$. We often denote the distribution with the probability density function f_θ where $\theta = (\boldsymbol{\mu}, \mathbf{\Sigma})$.

Definition 2.11 (likelihood)**.** Suppose we are given m observations $x_1, \ldots, x_m$ from a distribution with probability density function f_θ from a parametric family $\mathcal{P}$. The **likelihood** of parameters θ given observations $x_1, \ldots, x_m$ is equal to the probability of observing $x_1, \ldots, x_m$ given θ, that is:

$$\mathcal{L}(\theta \mid x_1, \ldots, x_m) = f_\theta(x_1, \ldots, x_m).$$

The idea of "likelihood" is to simply reverse the perspective: instead of assuming that the observations $x_1, \ldots, x_m$ have come from a distribution with parameters θ, we now assume that the parameters θ have actually come from given evidences $x_1, \ldots, x_m$.

Definition 2.12 (maximum likelihood estimation)**.** Suppose we are given m independent and identically distributed observations $x_1, \ldots, x_m$ from a distribution with probability density function f_θ from a parametric family $\mathcal{P}$.

The **maximum likelihood (ML) estimate** of θ, denoted as $\hat{\theta}$, are the parameters (represented as a vector) that maximize the likelihood function (or in practice, the log of the likelihood), that is:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\ \mathcal{L}(\theta \mid x_1, \ldots, x_m).$$

This method of estimating the parameters is called **maximum likelihood estimation (MLE)**.

MLE is a method of estimating the underlying patterns from a number of observations which are believed to have come from some kind of a pattern, e.g. the normal distribution. As mentioned before, this line of thought is essentially the same as solving the problem of learning. We will not go into details, but we will state an important result which we will utilize as an example later in the paper.

Example 2.13 (MLE for multivariate normal distribution)**.** Suppose we have m independent and identically distributed observations $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ of a random n-vector $\mathbf{X}$ that is assumed to have come from some n-variate normal distribution. The ML estimate of the parameters are given by

$$\hat{\boldsymbol{\mu}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \quad \text{and} \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{m}\sum_{i=1}^{m} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T.$$

Note that these are precisely the sample mean and covariance of the observations, which will be defined in the next section.

2.6. **Principal component analysis (PCA).**

Principal component analysis is a simple yet extremely useful application of eigendecomposition on a set of observations. First recall the following crucial result in linear algebra, due to spectral theorem:

Definition 2.14 (eigendecomposition)**.** Let $A \in \mathbb{R}^{n\times n}$ be a normal ($AA^T = A^TA$) matrix. The **eigendecomposition** (or **spectral decomposition**) of A is given by

$$A = VDV^T$$

where D is a diagonal matrix whose diagonals $\lambda_1 \geq \ldots \geq \lambda_n$ are the eigenvalues of A and V is an orthogonal matrix whose columns $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are the corresponding eigenvectors.

The goal of PCA is to find the "most significant" variations (directions) of a given set of observations (vectors). *PCA is powerful because it allows us to represent a number of (noisy) observations with only a few components.* Using the dot product, we formalize this concept by the notion of principal components:

Definition 2.15 (principal component)**.** Let $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ be m data points such that $\sum_{i=1}^m \mathbf{x}_i = 0$. Let X be an $n \times m$ matrix whose columns are $\mathbf{x}_1, \ldots, \mathbf{x}_m$.

The **(first) principal component** is defined as

$$\begin{aligned}\hat{\mathbf{w}}_1 &= \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}}\ \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i \cdot \mathbf{w})^2\\ &= \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}}\ \frac{1}{m}\ \mathbf{w}^T X X^T \mathbf{w}.\end{aligned}$$

For $2 \leq j \leq \min(m,n)$, we define the j**th principal component** as the principal component of $X^{(j)}$, whose columns $\mathbf{x}_k^{(j)}$ $(k = 1,\dots,m)$ are given by

$$\mathbf{x}_k^{(j)} = \mathbf{x}_k - \sum_{i=1}^{j-1}(\mathbf{x}_i \cdot \hat{\mathbf{w}}_i)\hat{\mathbf{w}}_i,$$

i.e., for each data point, we only leave its components orthogonal to all directions of previous principal components.

Note that we require the data points to be centered around the origin. In practice, we simply subtract the sample mean from each data point before using PCA.

We will now show how to compute the principal components. First recall the following definition:

Definition 2.16 (sample mean and sample covariance)**.** The **sample mean** of $\mathbf{x}_1,\dots,\mathbf{x}_m \in \mathbb{R}^n$ is defined as

$$\overline{\mathbf{x}} = \sum_{i=1}^{m}\mathbf{x}_i,$$

and the **sample covariance matrix** is defined as

$$Q = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T.$$

In particular, if the sample mean is zero, then $Q = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T = \dfrac{1}{m}XX^T$. Also, it is clear from the definition that Q is symmetric, which implies that we do have the eigendecomposition of Q.

Theorem 2.17 (PCA using the sample covariance matrix)**.** *Let $\mathbf{x}_1,\dots,\mathbf{x}_m \in \mathbb{R}^n$ be m data points such that $\overline{\mathbf{x}} = \sum_{i=1}^{m}\mathbf{x}_i = 0$. Let X be an $n \times m$ matrix whose columns are $\mathbf{x}_1,\dots,\mathbf{x}_m$.*

Let $Q = VDV^T$ be the eigendecomposition of the sample covariance matrix Q, with $D = \operatorname{diag}(\lambda_1,\dots,\lambda_m)$ and $V = (\mathbf{v}_1 \dots \mathbf{v}_m)$. Then, for $j = 1,2,\dots,\min(m,n)$,

$$\hat{\mathbf{w}}_j = \mathbf{v}_j$$

where $\hat{\mathbf{w}}_j$ is the jth principal component of $\mathbf{x}_1,\dots,\mathbf{x}_m$.

Proof. First, consider the case $j = 1$.

We noted earlier that $Q = \frac{1}{m}XX^T$. This means we may express the principal component as the following:

$$\begin{aligned}\hat{\mathbf{w}}_1 &= \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}}\ \frac{1}{m}\mathbf{w}^T XX^T\mathbf{w} \\ &= \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}}\ \mathbf{w}^T Q\mathbf{w} \\ &= \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}}\ \mathbf{w}^T(VDV^T)\mathbf{w} \\ &= \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}}\ (V^T\mathbf{w})^T D(V^T\mathbf{w}).\end{aligned}$$

Since V is orthogonal ($V^T = V^{-1}$), we may express $\hat{\mathbf{w}}_1$ as the following:

$$\begin{aligned}\hat{\mathbf{w}}_1 &= V\big(\underset{\|\mathbf{u}\|=1}{\operatorname{argmax}}\ \mathbf{u}^T D\mathbf{u}\big) \\ &= V\big(\underset{\|\mathbf{u}\|=1}{\operatorname{argmax}}\ \sum_{i=1}^{m}\lambda_i u_i^2\big).\end{aligned}$$

Since $\lambda_1 \geq \ldots \geq \lambda_m$, the vector $\hat{\mathbf{u}} = (1, 0, \ldots, 0)^T$ maximizes the above sum. Thus:

$$\begin{aligned}\hat{\mathbf{w}}_1 &= V(1, 0, \ldots, 0)^T \\ &= \mathbf{v}_1.\end{aligned}$$

For $j \geq 2$, note that we remove components in the direction of any of the previous eigenvectors. This means that the maximizing vector would simply be the next eigenvector $\mathbf{v}_j$. $\square$

Therefore, the principal components are simply the eigenvectors of the sample covariance matrix (which is again simply $Q = \frac{1}{m}XX^T$), even in the same order! This identification allows us to find the principal components by utilizing the existing methods that find eigenvalues and eigenvectors.

3. The Bhattacharyya kernel

In this section, we introduce the Bhattacharyya kernel, a novel kernel proposed by Jebara and Kondor in 2003. It is used when the input space is a set of distributions. First, we introduce the concept of the Bhattacharyya distance:

Definition 3.1 (A. Bhattacharyya, 1943)**.** Let p and q be continuous probability distributions over the same space Ω.

The **Bhattacharyya distance** between p and q is defined as

$$D_B(p, q) = -\ln BC(p, q),$$

where the **Bhattacharyya coefficient (BC)** between p and q is given by

$$BC(p, q) = \int_{\Omega} \sqrt{p(x)q(x)}dx.$$

Note that $0 \leq D_B \leq \infty$ and $0 \leq BC \leq 1$. The intuition behind the Bhattacharyya coefficient is that it measures the amount of *overlap* (i.e. similarity) between two distributions p and q, as it integrates (the square root of) their product over the whole space. This hints at the possibility that it can be used as a kernel between distributions.

Definition 3.2 (Jebara and Kondor, 2003)**.** Let $\mathcal{P}$ be the set of distributions over Ω. The **Bhattacharyya kernel on** $\mathcal{P}$ is the function $K_B : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ such that, for all $p, q \in \mathcal{P}$,

$$K_B(p,q) = BC(p,q) = \int_\Omega \sqrt{p(x)q(x)}dx.$$

By the same reasoning for the Bhattacharyya coefficient, this kernel is a measure of similarity. The following result is, indeed, crucial.

Theorem 3.3. *The Bhattacharyya kernel is symmetric and positive semi-definite.*

Proof. Symmetry is obvious.

For any $p_1, \dots, p_m \in \mathcal{P}$ and $\alpha_1, \dots, \alpha_m \in \mathbb{R}$,

$$\begin{aligned}\sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j K_B(p_i, p_j) &= \int_\Omega \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j \sqrt{p_i(x) p_j(x)} dx \\ &= \int_\Omega \left[\sum_{i=1}^m \left(\alpha_i \sqrt{p_i(x)} \right)^2 + \sum_{i \neq j} \alpha_i \alpha_j \sqrt{p_i(x) p_j(x)} \right] dx \\ &= \int_\Omega \left(\sum_{i=1}^m \alpha_i \sqrt{p_i(x)} \right)^2 dx \geq 0.\end{aligned}$$

Thus, K_B is PSD. □

Therefore, K_B *is* a kernel and we may utilize it within our learning algorithms, discuss its feature maps and feature spaces, and so on. Further, it has some nice properties, given that its inputs are probability distributions:

Remark 3.4.
(1) $0 \leq K_B \leq 1$.
(2) (the normalization property.) $K_B(p,p) = \int_\Omega p(x)dx = 1 \quad \forall p \in \mathcal{P}$.

4. The "bag of tuples" approach

At this point, the reader may still wonder if there ever exists an input space that is a set of distributions and not points in the first place, and if the Bhattacharyya kernel is ever useful. Here we introduce a useful way to represent observations where the Bhattacharyya kernel becomes handy.

Definition 4.1 (the "bag of tuples" representation)**.** In a learning problem, we say that each input is represented as a **"bag of tuples"** if it is a set of vectors, i.e. $\chi = \{\mathbf{x}_1, \dots, \mathbf{x}_m : \mathbf{x}_i \in \mathbb{R}^n\}$.

Recall that the most standard approach is to represent each input as a vector. But in many cases both the standard and the bag of tuples representation are possible:

Example 4.2.
(1) Suppose that each input in a learning problem is a monochromatic computer image of pixel size 20×20.

The standard way to represent it is to regard it as a vector in $\{0,1\}^{400}$, each coordinate containing information about whether or not a pixel is colored (1) or not (0).

A bag of tuples way to represent it is to regard it as a set of vectors in $\{1, 2, \ldots, 20\}^2$ such that each pixel, represented by its location $(x, y) \in \{1, 2, \ldots, 20\}^2$, is contained in the set if and only if it is colored.

(2) Now suppose that each input is a full-colored (with RGB components) image of the same size.

In the standard approach, each image is a vector, and now each coordinate would be a 3-digit 256-base number. For example, a pixel at position (10,5) with (R=100, G=255, B=10) would be represented by the number $100 \cdot 256^2 + 255 \cdot 256 + 10$ in the $(10 \cdot 20 + 5)$th coordinate of the image vector.

In the bag of tuples approach, each image is a set of vectors, where each vector is a 5-tuple (x, y, R, G, B). In the above example, the image itself is a set of vectors which contains the vector $(10, 5, 100, 255, 10)$.

Both are valid approaches to data. But the bag of tuples approach is particularly useful when *we want our kernel to have "soft invariance", e.g. invariance under translation and rotation.* In the standard vector representation, for example, we cannot capture similarity between the same pattern that arises in the first few coordinates in one image and in the last few coordinates (e.g. because the latter is a rotated version of the former one), because we only make comparisons in the coordinate level. In the bag of tuples representation, on the other hand, we can also compare the relative pixel positions so that we can capture such similarity.

5. The Bhattacharyya kernel between sets of vectors

Now that we demonstrated cases in which the "bag of tuples" approach can be useful, the remaining question is how we would define a similarity measure (kernel) in the bag of tuples representation, like the dot product in the standard vector representation. Obviously, there is no such thing as a dot product between sets of vectors. Instead, we extend the definition of the Bhattacharyya kernel as the following:

Definition 5.1 (Jebara and Kondor, 2003)**.** Let $\mathcal{X}$ be an input space of sets of vectors:

$$\mathcal{X} = \{\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \mid \mathbf{x}_i \in \Omega \subseteq \mathbb{R}^n, m \in \mathbb{N}\}.$$

Assume that each χ is a set of independent and identically distributed observations $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \Omega$ from an underlying distribution p that comes from a fixed parametric family $\mathcal{P}$ of distributions over Ω. Further assume that for each χ we have an estimate $\hat{p}$ of its underlying distribution p.

The **Bhattacharyya kernel on** $\mathcal{X}$ is a function $\overline{K}_B : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

$$\overline{K}_B(\chi, \chi') = K_B(\hat{p}, \hat{q}) = \int_\Omega \sqrt{\hat{p}(\mathbf{x})\hat{q}(\mathbf{x})} d\mathbf{x},$$

where $\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}, \chi' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{m'}\} \in \mathcal{X}$, and $\hat{p}$ and $\hat{q}$ are the estimated underlying distributions for χ and χ' respectively. Note that both $\hat{p}$ and $\hat{q}$ belong to the same parametric family $\mathcal{P}$.

This definition allows us to properly define a kernel to a learning problem in which the examples are represented as sets of vectors and we know an estimate of the distribution to each set beforehand. Thus there are two remaining questions:

how to estimate the distribution p (or its parameters) from each example χ, and how to actually compute the kernel for each pair of examples.

5.1. **The multivariate normal model using MLE.** We will consider the case in which $\mathcal{P}$ is assumed to be the set of n-variate normal distributions and utilize maximum likelihood estimation. This can be viewed as the most standard case which turns out to be useful in many scenarios. Thus, in the following, we let

$$\mathcal{P} = \{\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \mid \boldsymbol{\mu} \in \mathbb{R}^n, \boldsymbol{\Sigma} \in \mathbb{S}^n_{++}\}$$

where $\mathbb{S}^n_{++}$ is the set of n by n symmetric and positive definite (SPSD as well as non-degenerate) matrices.

Now we will use MLE to compute the values of the Bhattacharyya kernel.

Proposition 5.2. *With the conditions given by Definition 5.1 satisfied and with the above definition of $\mathcal{P}$, $\overline{K}_B(\chi, \chi')$ can be expressed in closed form for all $\chi, \chi' \in \mathcal{X}$, using MLE.*

Proof. Let $\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $\chi' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{m'}\} \in \mathcal{X}$. The conditions allow us to use MLE for each set of vectors: we know from Example 2.13 that the ML estimates are given by

$$\chi: \quad \hat{\boldsymbol{\mu}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \quad \text{and} \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T,$$

$$\chi': \quad \hat{\boldsymbol{\mu}'} = \frac{1}{m'}\sum_{i=1}^{m'} \mathbf{x}'_i \quad \text{and} \quad \hat{\boldsymbol{\Sigma}'} = \frac{1}{m'}\sum_{i=1}^{m'}(\mathbf{x}'_i - \hat{\boldsymbol{\mu}'})(\mathbf{x}'_i - \hat{\boldsymbol{\mu}'})^T.$$

Thus let $\hat{p} = \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$ and $\hat{q} = \mathcal{N}(\hat{\boldsymbol{\mu}'}, \hat{\boldsymbol{\Sigma}'})$ be the estimated underlying distributions of χ and χ' respectively. Then we can express the Bhattacharyya kernel between χ and χ' as

$$\begin{aligned}
\overline{K}_B(\chi, \chi') &= K_B(\hat{p}, \hat{q}) \\
&= \int_\Omega \sqrt{\frac{1}{(2\pi)^{n/2}|\hat{\boldsymbol{\Sigma}}|^{1/2}} e^{-(\mathbf{x}-\hat{\boldsymbol{\mu}})^T \hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}})/2}} \sqrt{\frac{1}{(2\pi)^{n/2}|\hat{\boldsymbol{\Sigma}'}|^{1/2}} e^{-(\mathbf{x}-\hat{\boldsymbol{\mu}'})^T \hat{\boldsymbol{\Sigma}'}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}'})/2}} d\mathbf{x} \\
&= (2\pi)^{-\frac{n}{2}} |\hat{\boldsymbol{\Sigma}}|^{-\frac{1}{4}} |\hat{\boldsymbol{\Sigma}'}|^{-\frac{1}{4}} \int_\Omega \exp\left(-\frac{1}{4}(\mathbf{x}-\hat{\boldsymbol{\mu}})^T \hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}) - \frac{1}{4}(\mathbf{x}-\hat{\boldsymbol{\mu}'})^T \hat{\boldsymbol{\Sigma}'}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}'})\right) d\mathbf{x} \\
&= |\boldsymbol{\Sigma}^*|^{\frac{1}{2}} |\hat{\boldsymbol{\Sigma}}|^{-\frac{1}{4}} |\hat{\boldsymbol{\Sigma}'}|^{-\frac{1}{4}} \exp\left(-\frac{1}{4}\hat{\boldsymbol{\mu}}^T \hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}} - \frac{1}{4}\hat{\boldsymbol{\mu}'}^T \hat{\boldsymbol{\Sigma}'}^{-1}\hat{\boldsymbol{\mu}'} + \frac{1}{2}\boldsymbol{\mu}^{*T}\boldsymbol{\Sigma}^*\boldsymbol{\mu}^*\right),
\end{aligned}$$

where $\boldsymbol{\mu}^* = \frac{1}{2}\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}} + \frac{1}{2}\hat{\boldsymbol{\Sigma}'}^{-1}\hat{\boldsymbol{\mu}'}$ and $\boldsymbol{\Sigma}^* = \left(\frac{1}{2}\hat{\boldsymbol{\Sigma}}^{-1} + \frac{1}{2}\hat{\boldsymbol{\Sigma}'}^{-1}\right)^{-1}$. This proves that $\overline{K}_B(\chi, \chi')$ can be computed in closed form for all $\chi, \chi' \in \mathcal{X}$ with the use of MLE. □

6. Extension to higher-dimensional spaces using the kernel trick

The aforementioned application of the Bhattacharyya kernel, despite of its name, does not use the idea of kernel trick. Recall from Section 2.3 that we may utilize kernels that come with feature maps to higher dimensional spaces, where the learning task (e.g. separation of points by a hyperplane) becomes possible. This is in many cases necessary - for example, if the inputs are monochromatic images represented as sets of vectors (x, y), then our current model finds a 2-dimensional Gaussian that depicts the prototypical pattern of the images, but if the image size is large and its patterns are more complicated then our model will not be able to capture the pattern successfully.

Thus, in order to bring a feature map to a higher-dimensional feature space, we now introduce a second kernel, this time on vectors, and fit a distribution in the feature space for each bag of (mapped) tuples.

Definition 6.1 (Jebara and Kondor, 2003)**.** Let $\mathcal{X} = \{\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \mid \mathbf{x}_i \in \Omega \subseteq \mathbb{R}^n, m \in \mathbb{N}\}$. Let $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a kernel and $\varphi : \mathbb{R}^n \to \mathcal{H}$ and $\mathcal{H}$ be a corresponding feature map and feature space respectively (Theorem 2.5) such that:

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}} \qquad \forall\ \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n.$$

Assume that each χ is a set of independent and identically distributed observations $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_m) \in \mathcal{H}$ from an underlying distribution p that comes from a fixed parametric family $\mathcal{P}_k$ of distributions over $\mathcal{H}$. Further assume that for each χ we have an estimate $\hat{p}$ of its underlying distribution p.

We define the **Bhattacharyya kernel on $\mathcal{X}$ using kernel** k as

$$\overline{K}_k(\chi, \chi') = K_k(\hat{p}, \hat{q}) = \int_{\mathcal{H}} \sqrt{\hat{p}(z)\hat{q}(z)}dz$$

where $\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}, \chi' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{m'}\} \in \mathcal{X}$ and $\hat{p}$ and $\hat{q}$ are the estimated underlying distributions of $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_m)$ and $\varphi(\mathbf{x}'_1), \ldots, \varphi(\mathbf{x}'_{m'})$ respectively. Note that $\hat{p}$ and $\hat{q}$ both belong to the same parametric family $\mathcal{P}_k$.

Basically, we map each tuple in the bag of tuples to an element of a high-dimensional feature space given by a new kernel k and fit a distribution on the feature space. This k could be the Gaussian RBF kernel, the polynomial kernel (Example 2.4), or any other SPSD function.

Now, we want to show that it is possible to compute the values using this kernel trick - that is, we want to show that what we only need to compute are the inner products $\langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}$, not the images $\varphi(\mathbf{x})$ and $\varphi(\mathbf{x}')$.

6.1. **Analogue of the normal model using MLE: regularized covariance form.**

Again, we will assume the most standard case and show how to compute the parameters efficiently with this feature map and feature space.

The biggest issue is to represent the analogous concepts in the large space $\mathcal{H}$. First, we utilize the Dirac notation.

Definition 6.2 (Dirac's bra-ket notation)**.** Let $\varphi : \mathbb{R}^n \to \mathcal{H}$ be a feature map where $\mathcal{H}$ is a Hilbert space (the feature space).

For $\mathbf{x} \in \mathbb{R}^n$, define the **ket** $|\mathbf{x}\rangle = \varphi(\mathbf{x})$. Also, define the **bra** $\langle \mathbf{x}| = \langle \varphi(\mathbf{x}), \cdot \rangle_{\mathcal{H}}$. The **inner product** in $\mathcal{H}$ between $|\mathbf{x}\rangle$ and $|\mathbf{x}'\rangle$ is simply denoted as $\langle \mathbf{x} \,|\, \mathbf{x}' \rangle$.

As its looks suggest, Dirac's notation divides up the inner product into two parts. Note that $\langle \mathbf{x}|$ the corresponding element of $|\mathbf{x}\rangle$ in the dual space according to Riesz representation theorem [5], thus justifying the notation for the inner product. This implies that, given a kernel k and a corresponding feature map φ and space $\mathcal{H}$ as in Theorem 2.5, the inner product *is* the kernel, i.e.

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

We may observe the use of kernel trick when we consider *the analogue of a real symmetric matrix and its corresponding linear map in* $\mathcal{H}$:

$$\mathbf{\Sigma} = \sum_{i=1}^{m} |\mathbf{x}_i\rangle \, \alpha_i \, \langle \mathbf{x}_i| \, .$$

Recall that symmetric matrices in $\mathbb{R}^n$, in particular SPSD matrices, have the form $\sum_{i=1}^m \alpha_i \mathbf{x}_i \mathbf{x}_i^T$ by eigendecomposition (Definition 2.14). Now, the corresponding linear map $\mathbf{\Sigma} : \mathcal{H} \to \mathcal{H}$ is given by

$$|\mathbf{x}\rangle \mapsto \left(\sum_i |\mathbf{x}_i\rangle \, \alpha_i \, \langle \mathbf{x}_i| \right) |\mathbf{x}\rangle = \sum_i |\mathbf{x}_i\rangle \, \alpha_i \, \langle \mathbf{x}_i \,|\, \mathbf{x}\rangle \, .$$

Then, $\langle \mathbf{x}_i \,|\, \mathbf{x}\rangle$ is just a real number, and if $\mathcal{H}$ is the feature space of a kernel then the inner product is simply the value of the kernel between $\mathbf{x}_i$ and $\mathbf{x}$.

The next step is to estimate a distribution p, or its parameters θ, to a bag of tuples $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_m) \in \mathcal{H}$. Again, we want p to be the analogue of the Gaussian. *In this paper, we will not discuss the case in which $\mathcal{H}$ is infinite-dimensional.* However, we note that even in the infinite case the Bhattacharyya kernel is computable, using the concept of Gaussian Processes (the infinite-dimensional analogue of multivariate Gaussians). For details, see [4].

For the rest of this paper, we assume that $\mathcal{H}$ is d-dimensional. We may define a d-dimensional normal distribution on $\mathcal{H}$:

Definition 6.3 (multivariate normal distribution on a Hilbert space)**.** Let $\varphi : \mathbb{R}^n \to \mathcal{H}$ be a feature map where $\mathcal{H}$ is a d-dimensional Hilbert space. A **d-dimensional normal distribution $\mathcal{N}(|\boldsymbol{\mu}\rangle, \mathbf{\Sigma})$ on** $\mathcal{H}$ is given by

$$p(|\mathbf{x}\rangle) = \frac{1}{(2\pi)^{d/2} |\mathbf{\Sigma}|^{1/2}} e^{-(\langle \mathbf{x}| - \langle \boldsymbol{\mu}|)\mathbf{\Sigma}^{-1}(|\mathbf{x}\rangle - |\boldsymbol{\mu}\rangle)/2}$$

where $|\boldsymbol{\mu}\rangle \in \mathcal{H}$ and $\mathbf{\Sigma}$ is a SPSD bilinear form of rank d.

However, there is a problem that calls for an alternative to the ML estimates. Note that the Gaussian RBF kernel is our favorite choice of kernel.

Remark 6.4. Assume the conditions in Definition 6.1. Let $k(\mathbf{x}, \mathbf{x}') = \exp(\|\mathbf{x} - \mathbf{x}'\| / 2\sigma_k^2)$ (the Gaussian RBF kernel) and let φ be the corresponding feature map so that $\varphi(\mathbf{x}) = |\mathbf{x}\rangle$. Let $\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be an example. Then, $|\mathbf{x}_1\rangle, \ldots, |\mathbf{x}_m\rangle$ span a subspace of dimension m in $\mathcal{H}$.

We will not prove this remark. However, it is important to notice that, because of this remark, *we are essentially fitting an m-dimensional normal distribution to m data points.* Such distribution is not very efficient in that we only want to capture the pattern of any given example. This problem is often called the problem of **overfitting**.

Because of this[1], Jebara and Kondor proposed a revised form of covariance that more effectively allows the extension to higher-dimensional spaces with the kernel k.

Definition 6.5 (regularized covariance form, Jebara and Kondor, 2003)**.** The **regularized covariance form** of the sample $|\mathbf{x}_1\rangle, \dots, |\mathbf{x}_m\rangle$ in $\mathcal{H}$ is defined as:

$$\mathbf{\Sigma}_{\text{reg}} = \sum_{l=1}^{r} |\mathbf{v}_l\rangle \, \lambda_l \, \langle \mathbf{v}_l| + \eta \sum_i |\mathbf{e}_i\rangle \langle \mathbf{e}_i| \,,$$

where $|\mathbf{v}_1\rangle, \dots, |\mathbf{v}_r\rangle$ and $\lambda_1, \dots, \lambda_r$ are the first r eigenvectors and eigenvalues of the ML estimated covariance (i.e. the sample covariance)

$$\hat{\mathbf{\Sigma}} = \frac{1}{m} \sum_{i=1}^{m} (|\mathbf{x}_i\rangle - |\hat{\boldsymbol{\mu}}\rangle)(\langle \mathbf{x}_i| - \langle \hat{\boldsymbol{\mu}}|) \quad \text{with} \quad \hat{\boldsymbol{\mu}} = \frac{1}{m} \sum_{i=1}^{m} |\mathbf{x}_i\rangle \,,$$

η is a regularization constant, and the $|\mathbf{e}_i\rangle$'s form an orthonormal basis for $\mathcal{H}$.

Note that $\sum_i |\mathbf{e}_i\rangle \langle \mathbf{e}_i|$ is just the analogue of the identity matrix, so that we are essentially adding a constant. Also note that using the first few largest eigenvalues/eigenvectors is a standard practice in finding "patterns", as we did in principal component analysis (PCA) in Section 2.6. In fact, in the next section we will introduce how to use PCA to obtain the eigenvectors in the regularized covariance form.

6.2. Kernel PCA.

In this section, we show how to obtain the first r largest eigenvalues and their corresponding eigenvectors using PCA, in the context of kernels and feature maps and without any explicit calculation of elements in the feature space. We also show how this deals with the problem of overfitting. The method was first developed in [6].

Theorem 2.17 showed that the principal components of the zero-centered data $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathbb{R}^n$ are precisely the eigenvectors of the sample covariance matrix $Q = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^T$, in the same order as in the size of the eigenvalues.

Kernel PCA is a generalization of PCA in the feature space. As before, let $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a kernel and $\varphi : \mathbb{R}^n \to \mathcal{H}$ be a corresponding feature map to the Hilbert space $\mathcal{H}$. We may first extend the definition of principal components given by Definition 2.15:

Definition 6.6 (principal component in the feature space)**.** Let $|\mathbf{x}_1\rangle, \dots, |\mathbf{x}_m\rangle \in \mathcal{H}$ be m data points such that $\sum_{i=1}^{m} |\mathbf{x}_i\rangle = 0$. The **(first) principal component** is defined as

$$|\hat{\mathbf{w}}_1\rangle = \operatorname*{argmax}_{\langle \mathbf{w} \,|\, \mathbf{w}\rangle = 1} \frac{1}{m} \sum_{i=1}^{m} \langle \mathbf{x}_i \,|\, \mathbf{w}\rangle^2 \,.$$

[1]In fact, Jebara and Kondor also showed that there is another huge problem due to the lack of alignment between the subspaces spanned by $\hat{\mathbf{\Sigma}}$ and $\hat{\mathbf{\Sigma}}'$ when we compare two normals, in the case when $\mathcal{H}$ is infinite-dimensional or when either $\hat{\mathbf{\Sigma}}$ or $\hat{\mathbf{\Sigma}}'$ is degenerate. This problem is also dealt with by the regularized covariance form given by Definition 6.5. In this paper, however, we will not discuss about these cases; see [4] for details.

For $2 \leq j \leq \min(m, \dim \mathcal{H})$, we define the **$j$th principal component** $|\hat{\mathbf{w}}_j\rangle$ as the principal component of the points

$$\left|\mathbf{x}_k^j\right\rangle = |\mathbf{x}_k\rangle - \sum_{i=1}^{j-1} \langle \mathbf{x}_i \mid \hat{\mathbf{w}}_i\rangle |\hat{\mathbf{w}}_i\rangle$$

for $k = 1, \ldots, m$.

Consequently, we obtain the following analogous result.

Theorem 6.7 (PCA using the sample covariance matrix in the feature space)**.** *Let $|\mathbf{x}_1\rangle, \ldots, |\mathbf{x}_m\rangle \in \mathcal{H}$ be m data points such that $\sum_{i=1}^m |\mathbf{x}_i\rangle = 0$. For $j = 1, \ldots, \min(m, \dim \mathcal{H})$, the jth principal component of $|\mathbf{x}_1\rangle, \ldots, |\mathbf{x}_m\rangle$ is the jth eigenvector of the matrix*

$$Q = \frac{1}{m} \sum_{i=1}^m |\mathbf{x}_i\rangle \langle \mathbf{x}_i| .$$

Proof. The proof is analogous to that of Theorem 2.17, but now with Definition 6.6. □

Kernel PCA relies on the following crucial observation.[2]

Corollary 6.8. *In the above, the eigenvectors $|\mathbf{v}_j\rangle$ lie in the span of $|\mathbf{x}_1\rangle, \ldots, |\mathbf{x}_m\rangle$.*

Proof. Since $|\mathbf{v}_j\rangle$ is an eigenvector to Q, we have

$$Q |\mathbf{v}_j\rangle = \lambda_j |\mathbf{v}_j\rangle .$$

Plugging in $Q = \dfrac{1}{m} \displaystyle\sum_{i=1}^m |\mathbf{x}_i\rangle \langle \mathbf{x}_i|$, we get

$$\frac{1}{m} \sum_{i=1}^m |\mathbf{x}_i\rangle \langle \mathbf{x}_i \mid \mathbf{v}_j\rangle = \lambda_j |\mathbf{v}_j\rangle ,$$

where $\langle \mathbf{x}_i \mid \mathbf{v}_j\rangle$ is just a real number. □

We now define what is often called the Gram matrix:

Definition 6.9 (Gram matrix)**.** The **Gram matrix** $K \in \mathbb{R}^{m\times m}$ of $|\mathbf{x}_1\rangle, \ldots, |\mathbf{x}_m\rangle$ is defined such that for $i, j = 1, \ldots, m$:

$$K_{ij} = \langle \mathbf{x}_i \mid \mathbf{x}_j\rangle .$$

Recall that for each i and j we have $\langle \mathbf{x}_i \mid \mathbf{x}_j\rangle = k(\mathbf{x}_i, \mathbf{x}_j)$, if the feature space is given by the kernel k. Now we can state the theorem.

Theorem 6.10 (Kernel PCA, Schölkopf et al., 1997)**.** *Let $|\mathbf{x}_1\rangle, \ldots, |\mathbf{x}_m\rangle \in \mathcal{H}$ be m data points such that $\sum_{i=1}^m |\mathbf{x}_i\rangle = 0$. The principal components $|\mathbf{v}_1\rangle, \ldots, |\mathbf{v}_r\rangle$ are related to the eigenvectors $\boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(r)}$ of the Gram matrix K by the following equation:*

$$|\mathbf{v}_l\rangle = \sum_{i=1}^m \alpha_i^{(l)} |\mathbf{x}_i\rangle \qquad (l = 1, \ldots, r)$$

where $\alpha_i^{(l)}$ is the ith coordinate of the m-dimensional vector $\boldsymbol{\alpha}^{(l)}$.

[2]In fact, an equivalent description of PCA in general is to start with this fact and analyze the covariance matrix given by the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$, or the kets of these. In this sense, PCA finds the largest covariance of the data points.

Proof. By Corollary 6.8, for each $l = 1, \dots, r$:

$$|\mathbf{v}_l\rangle = \sum_{i=1}^{m} \alpha_i^{(l)} |\mathbf{x}_i\rangle$$

for some $\alpha_1^{(l)}, \dots, \alpha_m^{(l)} \in \mathbb{R}$. Also, by Theorem 6.7, $|\mathbf{v}_l\rangle$ is the lth eigenvector to the sample covariance matrix

$$Q = \frac{1}{m} \sum_{i=1}^{m} |\mathbf{x}_i\rangle \langle \mathbf{x}_i| .$$

Then, the eigenvalue equation $Q |\mathbf{v}_l\rangle = \lambda_l |\mathbf{v}_l\rangle$ can be stated as the following:

$$\frac{1}{m} \sum_{i=1}^{m} |\mathbf{x}_i\rangle \langle \mathbf{x}_i| \sum_{j=1}^{m} \alpha_j^{(l)} |\mathbf{x}_j\rangle = \lambda_l \sum_{j=1}^{m} \alpha_j^{(l)} |\mathbf{x}_j\rangle .$$

Multiply $|\mathbf{x}_l\rangle$ to the left on both sides:

$$\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{m} \langle \mathbf{x}_l \,|\, \mathbf{x}_i\rangle \langle \mathbf{x}_i \,|\, \mathbf{x}_j\rangle \alpha_j^{(l)} = \lambda_l \sum_{i=1}^{m} \langle \mathbf{x}_l \,|\, \mathbf{x}_j\rangle \alpha_j^{(l)}.$$

Observe that this is equivalent to:

$$K^2 \boldsymbol{\alpha}^{(l)} = m\lambda_l K \boldsymbol{\alpha}^{(l)}$$

where K is the Gram matrix and $\boldsymbol{\alpha}^{(l)} = (\alpha_1^{(l)}, \dots, \alpha_m^{(l)})^T$. This is, of course, an m-dimensional eigenvalue/eigenvector problem

$$K\boldsymbol{\alpha}^{(l)} = (m\lambda_l)\boldsymbol{\alpha}^{(l)}.$$

□

There are two crucial consequences. First, we reduced an eigenvector problem in $\mathcal{H}$, whose dimension could be very large (even infinite), to an m-dimensional eigenvector problem.

Second and more importantly, *now we can compute the covariance without any explicit computation in $\mathcal{H}$ but with the inner products only.* These include $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i \,|\, \mathbf{x}_j\rangle$ as represented by the Gram matrix, as well as any new example $|\mathbf{x}\rangle$, whose projection onto the eigenvectors (often called *feature extraction*) can be expressed without any explicit computation in $\mathcal{H}$:

$$\langle \mathbf{x} \,|\, \mathbf{v}_l\rangle = \sum_{i=1}^{m} \alpha_i^{(l)} \langle \mathbf{x} \,|\, \mathbf{x}_i\rangle .$$

Therefore, we have demonstrated how kernel trick can be used in our process.

Finally, we will assume the following remark (see [6] for details):

Remark 6.11. In the case when $|\mathbf{x}_1\rangle, \dots, |\mathbf{x}_m\rangle$ are not centered, the above holds for $|\mathbf{x}_i^*\rangle = |\mathbf{x}_i\rangle - |\hat{\boldsymbol{\mu}}\rangle$.

6.3. **Summary of the process and computation of the Bhattacharyya kernel.** In this last section we will summarize the above process and show how to compute the Bhattacharyya kernel value.

First, to summarize the process:

(1) Start with two "bags of tuples" $\chi = \{\mathbf{x}_1, \dots, \mathbf{x}_m : \mathbf{x}_i \in \mathbb{R}^n\}$ and $\chi' = \{\mathbf{x}'_1, \dots, \mathbf{x}'_{m'} : \mathbf{x}'_i \in \mathbb{R}^n\}$.

(2) Given a kernel $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and a corresponding feature map $\varphi : \mathbb{R}^n \to \mathcal{H}$ to the feature space $\mathcal{H}$, for each bag of tuples $\chi = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ fit the normal distribution $\mathcal{N}(|\hat{\boldsymbol{\mu}}\rangle, \boldsymbol{\Sigma}_{\text{reg}})$. To compute $\boldsymbol{\Sigma}_{\text{reg}}$:

(a) Compute the r largest eigenvalues $\xi_1, \dots, \xi_r$ and eigenvectors $\boldsymbol{\alpha}^{(1)}, \dots, \boldsymbol{\alpha}^{(r)}$ of the Gram matrix $K \in \mathbb{R}^{m \times m}$: for $i, j = 1, \dots, m$,

$$K_{ij} = \langle \mathbf{x}_i \,|\, \mathbf{x}_j \rangle .$$

(b) Compute the r largest eigenvalues $\lambda_1, \dots, \lambda_r$ and the corresponding eigenvectors $|\mathbf{v}_1\rangle, \dots, |\mathbf{v}_r\rangle$ of the sample covariance matrix $\hat{\boldsymbol{\Sigma}}$ using Theorem 6.10: for $l = 1, \dots, r$,

$$\lambda_l = \frac{\xi_l}{m}$$

and

$$|\mathbf{v}_l\rangle = \sum_{i=1}^{m} \alpha_i^{(l)} |\mathbf{x}_i\rangle .$$

(c) Compute the regularized covariance form

$$\boldsymbol{\Sigma}_{\text{reg}} = \sum_{l=1}^{r} |\mathbf{v}_l\rangle \lambda_l \langle \mathbf{v}_l| + \eta \sum_i |\mathbf{e}_i\rangle \langle \mathbf{e}_i|$$

where η is a regularization constant and the $|\mathbf{e}_i\rangle$'s form an orthonormal basis for $\mathcal{H}$.

(d) In the case where $|\hat{\boldsymbol{\mu}}\rangle \neq |0\rangle$ (i.e. not centered) we may still follow the process because of Remark 6.11.

(3) Given two distributions $\hat{p} = \mathcal{N}(|\hat{\boldsymbol{\mu}}\rangle, \boldsymbol{\Sigma}_{\text{reg}})$ and $\hat{q} = \mathcal{N}(\left|\hat{\boldsymbol{\mu}}'\right\rangle, \boldsymbol{\Sigma}'_{\text{reg}})$ on $\mathcal{H}$, compute the Bhattacharyya kernel:

$$\overline{K}_k(\chi, \chi') = K_k(\hat{p}, \hat{q}) = \int_{\mathcal{H}} \sqrt{\hat{p}(z)\hat{q}(z)} dz.$$

We will finally note the following:

Remark 6.12. As in Proposition 5.2, for two distributions $p = \mathcal{N}(|\boldsymbol{\mu}\rangle, \boldsymbol{\Sigma})$ and $q = \mathcal{N}(|\boldsymbol{\mu}'\rangle, \boldsymbol{\Sigma}')$ on $\mathcal{H}$, the Bhattacharyya kernel value may be computed in closed form by the formula

$$\begin{aligned} K_k(p, q) = {} & |\boldsymbol{\Sigma}|^{-\frac{1}{4}} |\boldsymbol{\Sigma}'|^{-\frac{1}{4}} |\boldsymbol{\Sigma}^*|^{\frac{1}{2}} \\ & \exp\left(-\frac{1}{4} \langle \boldsymbol{\mu} | \boldsymbol{\Sigma}^{-1} |\boldsymbol{\mu}\rangle - \frac{1}{4} \langle \boldsymbol{\mu}' | \boldsymbol{\Sigma}'^{-1} |\boldsymbol{\mu}'\rangle - \frac{1}{2} \langle \boldsymbol{\mu}^* | \boldsymbol{\Sigma}^{*-1} |\boldsymbol{\mu}^*\rangle \right) \end{aligned}$$

where $|\boldsymbol{\mu}^*\rangle = \frac{1}{2}\boldsymbol{\Sigma}^{-1} |\boldsymbol{\mu}\rangle + \frac{1}{2}\boldsymbol{\Sigma}'^{-1} |\boldsymbol{\mu}'\rangle$ and $\boldsymbol{\Sigma}^* = \left(\frac{1}{2}\boldsymbol{\Sigma}^{-1} + \frac{1}{2}\boldsymbol{\Sigma}'^{-1} \right)^{-1}$.

This concludes the process of utilizing the Bhattacharyya kernel *and* the kernel trick. In their experiments with rotated/translated/scaled images, Jebara and Kondor show that the "bag of tuples" approach using the Bhattacharyya kernel performs better than the traditional approach. For details, see [4].

Acknowledgments. I thank my mentor Angela Wu for her earnest support and guidance throughout the REU program. I would also like to thank Professor Risi Kondor for his valuable suggestions on topics (in particular, this one) in machine learning.

References

[1] Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc*, 35(99-109), 4.

[2] Elmezain, M., Al-Hamadi, A., Rashid, O., & Michaelis, B. (2009). Posture and gesture recognition for human-computer interaction. *Advanced Technologies*, 415-439.

[3] Jebara, T., & Kondor, R. (2003). Bhattacharyya and expected likelihood kernels. In *Learning Theory and Kernel Machines* (pp. 57-71). Springer Berlin Heidelberg.

[4] Kondor, R., & Jebara, T. (2003). A kernel between sets of vectors. In *International Conference on Machine Learning* (pp. 361-368).

[5] Sally Jr, P. J. (2013). *Fundamentals of Mathematical Analysis* (Vol. 20). American Mathematical Soc..

[6] Schlkopf, B., Smola, A., & Mller, K. R. (1997). Kernel principal component analysis. In *Artificial Neural Networks–ICANN'97* (pp. 583-588). Springer Berlin Heidelberg.

[7] Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels* (pp. 366-369). The MIT Press.