Zero-shot classifier based on robust ellipsoid optimization

Kasper Rantamäki

School of Science

Bachelor's thesis
Espoo 21.07.2023

Supervisor and advisor

Prof. Kaie Kubjas

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the bachelor's thesis

Author Kasper Rantamäki

Title Zero-shot classifier based on robust ellipsoid optimization

Degree programme Bachelor's Programme in Science and Technology

Major Mathematics and Systems Sciences **Code of major** SCI3029

Teacher in charge and advisor Prof. Kaie Kubjas

Date 21.07.2023 **Number of pages** 37+1 **Language** English

Abstract

Zero-shot learning is an emerging problem setup within the broader field of machine learning, where the classifier is tasked with classifying points belonging to labels both seen and unseen in training. This addresses a prevalent problem in many applications of machine learning, the need for ever increasing amounts of data for training the models, by allowing the models to be trained on a dataset that does not contain data for all labels of interest.

In this thesis, we propose a novel approach for zero-shot learning. Our approach uses robustly trained ellipsoids that induce a distance metric in conjunction with kernel regression to find the label of a given point. This forms a mapping to an auxiliary space that can be chosen to contain additional information for the labels, which allows distinguishing between seen and unseen labels at test time.

Our zero-shot classifier was tested against the k-nearest neighbors classifier in two simple classification tasks — one with randomly generated 2-dimensional datapoints and the other with images of handwritten digits. Our model showed competitive results in cases where training data existed for all labels. Additionally, in the experiment with the 2-dimensional dataset with some labels left out of training our model was able to greatly outperform the k-nearest neighbors classifier, which is incapable of zero-shot classification.

Keywords Zero-shot learning, mathematical programming, support vector machine, kernel regression, machine learning

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Tekniikan kandidaatintyön tiivistelmä

Tekijä Kasper Rantamäki

Työn nimi Robustien ellipsoidien optimointiin pohjautuva nollan otoksen luokittelija

Koulutusohjelma Teknistieteellinen kandidaattiohjelma

Pääaine Matematiikka ja systeemitieteet **Pääaineen koodi** SCI3029

Vastuuopettaja ja ohjaaja Prof. Kaie Kubjas

Päivämäärä 21.07.2023 **Sivumäärä** 37+1 **Kieli** Englanti

Tiivistelmä

Nollan otoksen oppiminen (engl. zero-shot learning) on kasvava ongelmanasettelu laajemmalla koneoppimisen alalla, jossa luokittelijan tehtävänä on kategorisoida pisteitä sekä koulutuksessa nähtyihin, että näkemättömiin luokkiin. Tällä pyritään ratkaisemaan monissa koneoppimisen sovelluksissa ilmenevää ongelmaa: tarvetta jatkuvasti kasvavalle datamäärälle mallien kouluttamiseksi. Tämä mahdollistaa mallien kouluttaminen tietokannoilla, jotka eivät sisällä dataa kaikille halutuille luokille.

Tämä kandidaatintyö esittää uuden lähestymistavan nollan otoksen oppimiseen. Lähestymistapa hyödyntää sekä robustisti (engl. robustly) koulutettuja etäisyysmetriikan synnyttäviä ellipsoideja, että ydinregressiota (engl. kernel regression) löytääkseen luokan annetulle pisteelle. Tämä menetelmä luo kuvauksen apuavaruuteen, joka voidaan valita sisältämään lisätietoa luokista, mahdollistaen nähtyjen ja näkemättömien luokkien erottelun testivaiheessa.

Nollan otoksen luokittelijaamme verrattiin k-lähimmän naapurin luokittelijaan (engl. k-nearest neighbors classifier) kahdessa yksinkertaisessa luokittelutehtävässä: toisessa satunnaisesti luoduilla kaksiulotteisilla pisteillä ja toisessa käsinkirjoitettujen numeroiden kuvilla. Mallimme osoitti kilpailukykyisiä tuloksia tapauksissa, joissa koulutusdataa oli saatavilla kaikille luokille. Lisäksi testasimme malliamme kaksiulotteisilla pisteillä, jättämällä osan luokista pois koulutusdatasta, mutta sisällyttämällä ne testidataan. Tällöin mallimme suoriutui paremmin kuin k-lähimmän naapurin luokittelija, joka ei kykene ennustamaan koulutusdatattomia luokkia.

Avainsanat Nollan otoksen oppiminen, matemaattinen optimointi, tukivektorikone, ydinregressio, koneoppiminen

Contents

1 Introduction

The 21st century has witnessed a multitude of technological advancements, but few have been as remarkable as the strides made in machine learning. During the last few decades different machine learning models have accomplished human-like or beyond human abilities in variety of tasks like image classification [1, 2], text generation [3] and image synthesis [4, 5]. But machine learning as a field is not new, dating back in one form or another to the very early days of computing. McCulloch and Pitts considered an artificial neuron - the *McCulloch-Pitts neuron* - already back in 1943 [6], and this early work eventually paved the way for artificial neural networks, a highly influential machine learning paradigm.

The neural networks as we know them wouldn't come into being until 1986 when Rumelhart et. al. introduced the backpropagation algorithm for training the networks [7]. Still this leaves decades before their more general adoption for variety of tasks in the 2010s. While this lag could be contributed to the lack of computing power or just low interest in the field, in 2009 Jia Deng et. al. proposed that the reason could be the quality and quantity of the data available for training the models [8]. Thus, they compiled the *ImageNet* database - an extensive dataset of over 14 million images belonging to more than 20 thousand categories [8]. The dataset also sparked the *ImageNet Large Scale Visual Recognition Challenge*, where machine learning models from different teams could compete against one another on a 1000 category subset of the complete dataset [9]. Notable winners of the challenge are the *AlexNet* [2] from 2012, which standardized the use of convolutional neural networks (CNNs) for computer vision tasks and the use of graphical processing units (GPUs) for accelerating their training, and *ResNet* [1] from 2015, which prediction accuracy exceeded that of humans.

Whilst the increase in quantity of training data has allowed the models to accomplish human-like performance for example in aforementioned image recognition, the need for such quantities is very unhuman-like. Humans generally can learn to recognize new concepts from just a few examples and sometimes from no examples at all, but just a description. As a result, new fields have emerged within the realm of machine learning - namely the few-shot learning and the zero-shot learning - for exploring models making predictions from a few examples and from no examples respectively. Such models could be invaluable in applications were generating data is very expensive. One such application is in *neural language decoding*, where the label of stimulated concepts is retrieved from the neural data measured with e.g. functional magnetic resonance imaging (fMRI) or magnetoencephalography (MEG) methods [10, 11, 12].

This thesis proposes a model for zero-shot learning tasks and is composed as follows: firstly the concept of zero-shot learning will be discussed in more detail. This is followed by the study of a probabilistic interpretation that will lead the to problem formulation of the optimization task used in the training of the model. After discussing the training, the classification side of the model is covered followed by a more technical look into the implementation. Finally, the thesis concludes with the testing of the model on some example cases and general discussion.

2 Zero-shot learning

2.1 Basics

The idea of correctly recognizing a concept without an example might seem counter-intuitive, but is something humans can do quite naturally. As a classic example a human in general should be able to recognise a zebra if one has been told that zebra looks like a horse with black and white stripes. So while doable zero-shot prediction seems to require additional context provided in some form. This is indeed what Palatucci et. al. propose in their seminal paper [13] as a *semantic output code classifier*. This is formally defined in Definition 1.

Definition 1. *A semantic code classifier is a mapping $\mathcal{H} : F^n \to L$ defined as the composite of two functions - first mapping to a semantic space and the other from semantic space to label space - of form:*

$$\begin{aligned} \mathcal{H} &= \mathcal{L}(\mathcal{S}(\cdot)) \\ \mathcal{S} &: F^n \to S^p \\ \mathcal{L} &: S^p \to L \end{aligned}$$

where F^n is the feature space, S^p is the semantic space and L is the label space.

Depending on the problem at hand the feature space F^n could consist of the pixel values of images that are to be classified or time varied measurements from the sensors of a MEG machine to name a few examples. Label space L usually just consists of the names of the categories to which the datapoints are classified into.

The semantic space S^p is then the tool for providing the additional context for the model. For the semantic space to be valid the vectors should be one-hot encodings of the labels that retain the semantic and syntactic information of the underlying label. One example of such vectors are the *word2vec* vectors [14, 15], which have become a standard in zero-shot learning [16, 17].

Indeed, the word2vec vectors maintain a very intuitive relation between the concepts. This is evidently shown in some simple arithmetic examples like:

$$\text{“brother”} - \text{“man”} + \text{“woman”} = \text{“sister”}$$

That is when one subtracts the vector for word “man” from the vector for word “brother” and adds the vector for word “woman” the labeled vector closest to the resulting one is the vector for word “sister”.

Currently the field of zero-shot learning is very much in its infancy and thus a large variety of different algorithms have been proposed, with differing levels of success. Most of these still commonly contain some regression algorithm for the mapping $\mathcal{S}$ and basic nearest neighbour search for mapping $\mathcal{L}$. In turn some more exotic approaches utilize specialized auto-encoder models [18, 19]. For a more in-depth overview of the field see [16, 17].

2.2 Overview of our algorithm

Our algorithm takes a somewhat different approach. Instead of directly training a regression algorithm from the feature space to the semantic space, our model finds ellipsoids around the datapoints of each label used in training. The purpose of the ellipsoids is to model some unknown probability distributions that the datapoints are likely to follow. This is covered in more detail in Section 3 and Section 4.

The definition of an ellipsoid induces a metric, which can be thought of as the squared Euclidean distance in a space linearly transformed by some characteristic matrix. The distances from each ellipsoid as given by this metric are used in kernel regression to form the mapping $\mathcal{S}$, which can be considered as a weighted average of the semantic vectors associated with each label. The mapping $\mathcal{L}$ is in turn just a trivial nearest neighbor search. This classification framework is covered in depth in Section 5.

3 Basic definitions

3.1 Probabilistic interpretation

Consider that we want to model the behaviour of points in some set X. Each point consists of features, which by their nature follow some unknown probability distributions. Clearly then, if each feature follows its own distribution we can model the set X with a vector of random variables $\mathbf{X} = [X^{(1)} \quad X^{(2)} \quad \dots \quad X^{(n)}]^T$. The dependencies of the random variables in $\mathbf{X}$ can be studied with the covariance matrix $K_{\mathbf{XX}}$:

Definition 2. *A covariance matrix $K_{\mathbf{XX}} \in \mathbb{R}^{n \times n}$ of a vector of random variables $\mathbf{X}$ is a symmetric matrix such that at:*

$$(K_{\mathbf{XX}})_{ij} = (K_{\mathbf{XX}})_{ji} = \mathrm{Cov}[X^{(i)}, X^{(j)}] \; \forall \, i, j \in \{1, \; 2, \; \dots, \; n\}$$

Identically this can be expressed as:

$$K_{\mathbf{XX}} = \mathrm{Cov}(\mathbf{X}, \mathbf{X}) = \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^T] \tag{1}$$

Due to the properties of covariance we further know some properties of the covariance matrix $K_{\mathbf{XX}}$, like crucially that it is a symmetric and positive semi-definite:

Theorem 3. *Any given covariance matrix $K_{\mathbf{XX}}$ is symmetric positive semi-definite.*

Proof. Symmetricity of the covariance matrix is a result of the commutative property of the covariance function. A given symmetric matrix $K_{\mathbf{XX}} \in \mathbb{R}^{n \times n}$ is positive semi-definite iff:

$$\mathbf{v}^T K_{\mathbf{XX}} \mathbf{v} \geq 0 \;\forall\; \mathbf{v} \in \mathbb{R}^n \tag{2}$$

Using Equation (1) and any arbitrary vector $\mathbf{v} \in \mathbb{R}^n$ we get:

$$\begin{aligned} \mathbf{v}^T K_{\mathbf{XX}} \mathbf{v} &= \mathbf{v}^T \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^T]\mathbf{v} \\ &= \mathbb{E}[\mathbf{v}^T(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^T\mathbf{v}] \\ &= \mathbb{E}[(\mathbf{v}^T(\mathbf{X} - \mathbb{E}[\mathbf{X}]))^2] \geq 0 \end{aligned}$$

□

With some additional assumptions we can make sure the covariance matrix is not only positive semi-definite, but positive definite. Before looking into such assumptions it is worthwhile to consider some properties of positive definite matrices. We will begin this discussion with the following lemma.

Lemma 4. *A given matrix $A \in \mathbb{R}^{n \times n}$ is symmetric positive definite iff it has strictly positive eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$.*

Proof. We will prove this statement in two parts.

Part 1. If a given symmetric matrix A has strictly positive eigenvalues it is positive definite:
By spectral theorem for symmetric matrices with positive eigenvalues, must exist a decomposition of form:

$$A = Q^T \Lambda Q \tag{3}$$

where Q is an orthogonal matrix and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. As Q is orthogonal and thus defines a pure rotation we have:

$$\mathbf{w} := Q\mathbf{v} \neq \mathbf{0} \;\forall\; \mathbf{v} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \tag{4}$$

By definition of a positive definite matrix holds:

$$\mathbf{v}^T A\mathbf{v} > 0 \;\forall\; \mathbf{v} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \tag{5}$$

Placing Equations (3) and (4) into Equation (5) we find:

$$\mathbf{v}^T A\mathbf{v} = \mathbf{w}^T \Lambda \mathbf{w} = \sum_{i=1}^{n} \lambda_i w_i^2 > 0$$

By Equation (4) we know that $w_i^2 > 0$ for all $i \in \{1, \ldots, n\}$, which means that the eigenvalues λ_i for all $i \in \{1, \ldots, n\}$ must be strictly positive. □

Part 2. If a given symmetric matrix A is positive definite it has strictly positive eigenvalues: Firstly, as a proof by contradiction assume that a positive definite matrix A has a zero eigenvalue λ. Then would hold:

$$\exists\; \mathbf{v} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \;:\; A\mathbf{v} = \lambda\mathbf{v} = \mathbf{0}$$

Placing the eigenvector $\mathbf{v}$ associated with the zero eigenvalue λ into Equation (5) would give:

$$\mathbf{v}^T A\mathbf{v} = \mathbf{v}^T\mathbf{0} = 0$$

contradicting the definition of a positive definite matrix in Equation (5).

Secondly, as a proof of contradiction assume that a positive definite matrix has a negative eigenvalue λ with an associated eigenvector $\mathbf{v}$. We find:

$$\mathbf{v}^T A\mathbf{v} = \mathbf{v}^T(\lambda\mathbf{v}) = \mathbf{v}^T\mathbf{v}\lambda \leq 0 \tag{6}$$

as always must hold that $\mathbf{v}^T\mathbf{v} \geq 0$. This again contradicts the definition of a positive definite matrix in Equation (5).

Thus, if a matrix is positive definite it must have positive eigenvalues. □

We can use the results of Lemma 4 to show that when a covariance matrix is invertible it is also positive definite as proven in Theorem 5.

Theorem 5. *A symmetric positive semi-definite covariance matrix $K_{\mathbf{XX}} \in \mathbb{R}^{n\times n}$ is symmetric positive definite iff it is invertible:*

Proof. We will prove this statement in two parts. However, first note that by the adjugate form of the inversion:

$$K_{\mathbf{XX}}^{-1} = \frac{1}{\det(K_{\mathbf{XX}})}\mathrm{adj}(K_{\mathbf{XX}}) \tag{7}$$

for any given (square) matrix to be invertible it must have non-zero eigenvalues as otherwise the determinant will become zero making the inverse undefined. This is a property that we will use in both parts of the proof.

Part 1. If a covariance matrix $K_{\mathbf{XX}} \in \mathbb{R}^{n\times n}$ is symmetric positive definite it is invertible:
As laid out in Equation (7) for matrix to be invertible it must have non-zero eigenvalues. However, by Lemma 4 we already know that a symmetric positive definite matrix will always have strictly positive eigenvalues, which are thus always non-zero, concluding the proof. □

Part 2. If a symmetric positive semi-definite covariance matrix $K_{\mathbf{XX}} \in \mathbb{R}^{n\times n}$ is invertible it is positive definite:
The definition of a positive semi-definite matrix is laid out in Equation (2). We can use it to study the properties of the associated eigenvalues. Let $\mathbf{v}$ be an arbitrary eigenvector of $K_{\mathbf{XX}}$. Then:

$$\mathbf{v}^T K_{\mathbf{XX}} \mathbf{v} = \mathbf{v}^T(\lambda \mathbf{v}) = \mathbf{v}^T \mathbf{v} \lambda \geq 0 \tag{8}$$

As always holds that $\mathbf{v}^T\mathbf{v} \geq 0$ then must also hold that $\lambda \geq 0$. As the choice of eigenvector $\mathbf{v}$ was arbitrary, must positive semidefinite matrix always have strictly non-negative eigenvalues. By the adjugate form of inversion an invertible matrix has only non-zero eigenvalues. Hence, a invertible positive semi-definite matrix would have strictly positive eigenvalues. From Lemma 4 we know this is exactly the definition of a positive definite matrix, concluding the proof. □

3.2 Modelling from samples and minimum volume ellipsoids

All of the theory covered in Section 3.1 provides us a framework for defining the region containing all points in X that best models the probability distributions in $\mathbf{X}$. Indeed, if the covariance matrix $K_{\mathbf{XX}} \in \mathbb{R}^{n\times n}$ is positive definite we can use it to define an ellipsoid:

Definition 6. *An ellipsoid in $\mathbb{R}^n$ centered at $\mathbf{m} \in \mathbb{R}^n$ with a positive definite characteristic matrix $A \in \mathbb{R}^{n\times n}$ is given by the set of points:*

$$\mathcal{E}_A(\mathbf{m}) = \{\mathbf{x} \in \mathbb{R}^n : (\mathbf{x} - \mathbf{m})^T A (\mathbf{x} - \mathbf{m}) = 1\} \tag{9}$$

The characteristic matrix of an ellipsoid carries a lot of useful information as the eigenvectors of it are the principal axis of the ellipsoid and the eigenvalues the semi-axis lengths as proven in Theorem 7.

Theorem 7. *The principal axes of an ellipsoid are the eigenvectors of the characteristic matrix and the semi-axes lengths are the reciprocals of the squares of the eigenvalues of the* *characteristic matrix.*

Proof. Consider an ellipsoid centered at origin with a characteristic matrix $A \in \mathbb{R}^{n\times n}$. As the characteristic matrix must be positive definite it has an eigendecomposition of form:

$$A = Q\Lambda Q^T \tag{10}$$

where $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix with the eigenvectors of A as its columns and $\Lambda \in \mathbb{R}^{n\times n}$ is a diagonal matrix with the eigenvalues of A on its diagonal. With the eigendecomposition we can rewrite the ellipsoid equation defined in Equation (9) as:

$$\begin{aligned}\mathcal{E}_A(\mathbf{0}) &= \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^T A\mathbf{x} = 1\} \\ &= \{\mathbf{x} \in \mathbb{R}^n : (Q^T\mathbf{x})^T \Lambda Q^T \mathbf{x} = 1\} \\ &= \{\mathbf{y} \in \mathbb{R}^n : \mathbf{y}^T \Lambda \mathbf{y} = 1\}\end{aligned}$$

where $\mathbf{y} = q(\mathbf{x}) = Q^T\mathbf{x}$. As Q is by definition orthogonal matrix, the mapping q describes a pure rotation of the axes of the coordinate system. The new axes - the principal axes of the ellipsoid - are then defined by the eigenvectors of A. The general ellipsoid equation for n-dimensional ellipsoid is defined as:

$$\mathbf{y}^T\Lambda\mathbf{y} = \sum_{i=1}^{n} \lambda_i y_i^2 = \sum_{i=1}^{n} \frac{y_i^2}{l_i^2} = 1$$

where the semi-axes lengths l_i are the reciprocals of the squares of the eigenvalues $\lambda_i = l_i^{-2}$. Same extends w.l.o.g. to ellipsoids centered at an arbitrary point $\mathbf{m} \in \mathbb{R}^n$. □

Theorem 7 outlines that the size of the ellipsoid scales inversely to the characteristic matrix A. For example in the case of a covariance matrix, for which greater values as elements describe a greater spread of the datapoints and hence a requirement for a larger ellipsoid, this is undesirable. However, this can be easily fixed by using the inverse of the covariance matrix as the characteristic matrix since by the eigendecomposition defined in Equation (10) we get:

$$A = K_{\mathbf{XX}}^{-1} = (Q\Lambda Q^T)^{-1} = Q\Lambda^{-1}Q^T$$

where as Λ is diagonal holds:

$$\operatorname{diag}(\lambda_1, \lambda_2, ..., \lambda_n)^{-1} = \operatorname{diag}(\lambda_1^{-1}, \lambda_2^{-1}, ..., \lambda_n^{-1})$$

Knowing the properties of scaling the characteristic matrix we could define the smallest possible ellipsoid around a given set of points X. We can formulate the *minimum volume ellipsoid* as an optimization problem defined in Definition 8.

Definition 8. *Minimum volume ellipsoid (MVE) is the ellipsoid with the smallest volume that still covers all wanted points. This can be formulated as an optimization problem:*

$$\begin{aligned} \min_{A} \quad & V(\mathcal{E}_A(\mathbf{m})) \\ \text{s.t.} \quad & (\mathbf{x}-\mathbf{m})^T A(\mathbf{x}-\mathbf{m}) \leq 1 \; \forall \, \mathbf{x} \in X \end{aligned} \tag{11}$$

where $V(\cdot)$ returns the volume of a given input.

In the case where the covariance matrix $K_{\mathbf{XX}}$ is known and positive definite we could make the assumption that it models the spread of the datapoints in the set X the best. Thus, instead optimizing for the whole matrix A we can exchange it into the problem defined in Definition 8 as $A = kK_{\mathbf{XX}}^{-1}$ where $k \in \mathbb{R}$ is a scaling coefficient. Then it would be enough to find the optimal coefficient k.

4 Model definition and training

In Section 3 we only consider a single set of points X, which can be assumed to belong to some given label, but for classifiers to be at all useful, they need to be able to distinguish datapoints belonging to at least two separate labels. Furthermore, as the point of zero-shot classifiers, as outlined in Section 2, is to extend the classification abilities beyond the limitations of current models we generally want to consider a set of distinct labels L_0.

However, for simplicity's sake consider that the set of training points F from all labels $l \in L_0$ is split into two sets X and Y, such that points $\mathbf{x} \in X$ share the same label $l_x \in L_0$ and points $\mathbf{y} \in Y$ have some other label $l_y \in L_0 \setminus \{l_x\}$. This split only provides the framework for a binary classifier, but it is extendable into broader cases as discussed later in Section 5.

Even though minimum volume ellipsoids defined in Definition 8 can be used to form a classifier on their own, as they only consider points in a single set, the classifier would not be all that robust. Robustness is crucial as we work with data that is stochastic in nature. Thus, instead of finding the ellipsoid with the least volume, we would like to find the ellipsoid of which surface is the furthest distance from both the points in X and the points in Y, while enclosing as many points in X as possible. This can be alternatively formulated as finding the ellipsoid that maximizes the distance to the decision boundary for the closest points in both sets X and Y.

This is very similar to the definition for the *support vector machine* (SVM) algorithm [20, 21], with only the slight difference that we are attempting to find an ellipsoid rather than a hyperplane to separate the sets of points. And indeed we can create an algorithm inspired by the support vector machine algorithm to fit our purpose.

4.1 Related works

Our algorithm is just one (naive) way of finding an ellipsoidal boundary in support vector machine like fashion. A more common alternative would be to use what are called kernel machines, which operate in a higher dimensional space implicit feature space, where a hyperplane could be found to separate the sets of points while retaining that in the original space the separation is in form of an ellipsoid (or any nonlinear boundary for that matter). See for example [22, 23] for some examples of this approach.

4.2 Modified support vector machine algorithm

By basic principle we would like for the classifier to provide one signal for all datapoints in X and another for all datapoints in Y. Considering the ellipsoid $\mathcal{E}_A(\mathbf{m})$ defined in Definition 6 if it is chosen so that it encompasses all points $\mathbf{x} \in X$, but not $\mathbf{y} \in Y$ would hold:

$$(\mathbf{x}-\mathbf{m})^T A(\mathbf{x}-\mathbf{m}) - 1 \leq 0 \quad \text{and} \quad (\mathbf{y}-\mathbf{m})^T A(\mathbf{y}-\mathbf{m}) - 1 \geq 0 \tag{12}$$

But to optimize for A requires some way of penalizing the wrong classifications and rewarding the correct ones. Firstly, we will want to introduce a buffer zone to add some robustness by rewriting equations in (12) in form:

$$(\mathbf{x}-\mathbf{m})^T A(\mathbf{x}-\mathbf{m}) - 1 \leq -1 \quad \text{and} \quad (\mathbf{y}-\mathbf{m})^T A(\mathbf{y}-\mathbf{m}) - 1 \geq 1 \tag{13}$$

Then we can relax this hardened constraint by introducing non-negative variables u_1, ..., $u_{|X|}$ and v_1, ..., $v_{|Y|}$ turning the equations from (13) into form:

$$\begin{aligned}(\mathbf{x}_i-\mathbf{m})^T A(\mathbf{x}_i-\mathbf{m}) &\leq u_i, &\quad \forall i \in \{1, ..., |X|\}\\ (\mathbf{y}_i-\mathbf{m})^T A(\mathbf{y}_i-\mathbf{m}) - 2 &\geq -v_i, &\quad \forall i \in \{1, ..., |Y|\}\end{aligned}$$

The variables u_i and v_i measure how far from the correct classification the corresponding datapoints $\mathbf{x}_i$ and $\mathbf{y}_i$ fall. One can get an intuition for this by noting that when any given u_i or v_i is one then we have returned from the buffered definition in (13) to the non-buffered one in (12). And identically if the given u_i or v_i is greater than one the classification will be wrong. Thus, our objective would be to minimize the sum of these measures. This gives us a problem formulation of form:

$$\begin{aligned} \min_{A} \quad & \sum_{i=1}^{|X|} u_i + \sum_{i=1}^{|Y|} v_i \\ \text{s.t.:} \quad & u_i - (\mathbf{x}_i - \mathbf{m})^T A (\mathbf{x}_i - \mathbf{m}) \geq 0, && \forall i \in \{1, ..., |X|\} \\ & v_i + (\mathbf{y}_i - \mathbf{m})^T A (\mathbf{y}_i - \mathbf{m}) - 2 \geq 0, && \forall i \in \{1, ..., |Y|\} \\ & u_i \geq 0, && \forall i \in \{1, ..., |X|\} \\ & v_i \geq 0, && \forall i \in \{1, ..., |Y|\} \end{aligned} \tag{14}$$

The used centerpoint $\mathbf{m} \in \mathbb{R}^n$ can be found as the the arithmetic mean $\mathbf{m} = \bar{\mathbf{x}} \in \mathbb{R}^n$ of the datapoints $\mathbf{x} \in X$. Also note that additional constraints may need to be added to keep A symmetric positive definite.

To evaluate the performance and convergence of possible applied optimization solvers we should know more about the nature of the problem at hand. And indeed we can show that the problem is convex:

Theorem 9. *The optimization problem defined in Equation* (14) *is convex.*

Proof. Values u_i and v_i can be considered as affine functions and are thus both convex and concave. As a sum of convex functions the objective function is convex. The constraints are also sums of affine functions and thus convex, meaning that the problem is a convex *linear* optimization problem as it consists of optimizing a linear function over a convex set with linear boundaries. □

4.3 Model training

The above problem formulation does not yet make it clear how we can arrive into a zero-shot classifier, but it is important to remind oneself about the definitions of the sets X and Y as only thing we care is that the points in set X share the same label l_x of interest while Y contains all other points with labels $l_y \in L_0 \setminus \{l_x\}$. Here we also make a crucial distinction for zero-shot learning between the set of distinct initial labels L_0 for which we have training data and the total set of distinct labels L. Obviously, it must then hold that $L_0 \subset L$. Using these definitions we can divide the set of training points F into sets X and Y for any arbitrary choice of $l \in L_0$ and thus optimize the ellipsoid according to Equation (14) for all labels independently. This also means that this training is massively parallelizable.

5 Classification

The problem with allocating only closed regions of the feature space for the labels is that if we considered the class based on the region in which the point falls, we would end up with $|L_0| + 1$ possible classifications (assuming the ellipsoids don't overlap), while only having $|L_0|$ labels. Thus, the training does not find an unambiguous mapping itself. Independent of whether we want to make zero-shot or generic predictions we would like to use as much of the information we have found in the training phase and we would like for the classifier to require minimal change when doing generic classifications instead of the zero-shot ones. In all such cases the classification would require finding some form of approximation for the label in question.

5.1 Zero-shot classification

In Section 3.1 we defined a random vector $\mathbf{X}$ to model the probability distributions of the points in set X. Now consider similarly that we have a random vector $\mathbf{S} = [S^{(1)} \quad S^{(2)} \quad \ldots \quad S^{(p)}]^T$ which can take the value of the semantic vectors $\mathbf{s} \in S$ and a random variable C that likewise can take the value of the labels (classes) of the datapoints. The labels of the datapoints can be accessed with some function $c(\cdot)$ such that $c(\mathbf{f}) \in L_0$ for all $\mathbf{f} \in F$. Here F is again the set of training points. Additionally, we want to define a set of semantic vectors S_0 which correspond to the labels in the set L_0 while the set of all semantic vectors S understandably has vectors for all of the labels in set L. The mapping $\mathcal{S}$ from Definition 1 can then be considered as a conditional expectation:

$$\mathcal{S}(\mathbf{f}) = \mathbb{E}[\mathbf{S}|C = c(\mathbf{f})] \tag{15}$$

Then according to Theorem 10 we can approximate the mapping from Equation (15) as a weighted average of the semantic vectors $\mathbf{s} \in S_0$:

Theorem 10. *The mapping $\mathcal{S}$ defined as conditional expectation in equation (15) can be approximated by a weighted average.*

Proof. Consider the conditional expectation for a single random variable $S^{(i)}$ in (discrete) random vector $\mathbf{S}$ given by:

$$\begin{aligned}
\mathcal{S}(\mathbf{f})_i &= \mathbb{E}[S^{(i)}|C = c(\mathbf{f})] \\
&= \sum_{\mathbf{s} \in S_0} s_i \mathbb{P}[\mathbf{S} = \mathbf{s}|C = c(\mathbf{f})] \\
&= \sum_{\mathbf{s} \in S_0} s_i \frac{\mathbb{P}[\mathbf{S} = \mathbf{s} \cap C = c(\mathbf{f})]}{\mathbb{P}[C = c(\mathbf{f})]}
\end{aligned}$$

where s_i is the i-th element in a given vector $\mathbf{s} \in S_0$ and the formula for conditional probability $\mathbb{P}[A|B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}$ was used. As this holds for any arbitrary random variable

$S^{(i)}$ we find:

$$\mathcal{S}(\mathbf{f}) = \mathbb{E}[\mathbf{S}|C = c(\mathbf{f})] = \sum_{\mathbf{s}\in S_0} \frac{\mathbb{P}[\mathbf{S} = \mathbf{s} \cap C = c(\mathbf{f})]\mathbf{s}}{\mathbb{P}[C = c(\mathbf{f})]} \tag{16}$$

$\square$

Using kernel density estimation [24, 25] we can approximate the probabilities in Equation (16) and work it into essentially the form of Nadaraya-Watson kernel regression [26, 27]:

$$\mathcal{S}(\mathbf{f}) = \frac{\sum_{\mathbf{s}\in S_0} K(d_{\mathbf{s}}(\mathbf{f}))\mathbf{s}}{\sum_{\mathbf{s}\in S_0} K(d_{\mathbf{s}}(\mathbf{f}))} \tag{17}$$

where K is the kernel function and $d_{\mathbf{s}}$ is the distance function from the label associated with the semantic vector $\mathbf{s}$.

What we have derived here is in its simplicity a weighted average of the semantic vectors of which labels we had in training. The weight for each semantic vector is dependent on the distance from the point to their respective ellipsoids as given by some metric. Additionally, the distance is passed through some user defined function, which for example can provide a nonlinear dependency between the distance and the final weight.

To make this work we must firstly define a distance metric. One intuitive metric would be the shortest Euclidean distance from each ellipsoid to the point. However, as the definition of the ellipsoid in Definition 6 itself induces a metric it would be a far more natural choice. This is actually the squared *Mahalanobis distance* [28] from a distribution approximated by the inverse of the characteristic matrix A:

Definition 11. *Given a probability distribution $\mathbf{X} \in \mathbb{R}^n$ with mean $\mathbf{m} \in \mathbb{R}^n$ and a positive definite covariance matrix $K_{\mathbf{XX}} \in \mathbb{R}^{n\times n}$ the Mahalanobis distance to some arbitrary point $\mathbf{v} \in \mathbb{R}^n$ from $\mathbf{X}$ is:*

$$d_M(\mathbf{v},\ \mathbf{X}) = \sqrt{(\mathbf{v} - \mathbf{m})^T K_{\mathbf{XX}}^{-1}(\mathbf{v} - \mathbf{m})}$$

If we use the notation from Equation (12) we will end up with negative values for points within the ellipsoid and positive for those outside of it meaning that the kernel function K should be chosen to account for this. For example one choice could be maximum differences computed from a vector of distances $\mathbf{d} = [d_1 \quad \dots \quad d_{|L_0|}]^T$ as:

$$\mathbf{w} = [\max(\mathbf{d}) - d_1 \quad \dots \quad \max(\mathbf{d}) - d_{|L_0|}]^T \tag{18}$$

To create a nonlinear dependency for the weights and the distance, the maximum differences computed in Equation (18) could be passed through an exponential function. With the weighted average computation the exponentials essentially form into the *softmax* function as defined in Definition 12.

Definition 12. *The softmax function is a generalization of logistic function and can be used to convert a vector of real values* $\mathbf{z} \in \mathbb{R}^m$ *into a probability distribution of* m *outcomes i.e.* $\sigma : \mathbb{R}^m \to (0,1)^m$*:*

$$\sigma(\mathbf{z})_i = \frac{\exp z_i}{\sum_{j=1}^{m} \exp z_j} \quad \forall i \in \{1, ..., m\}$$

Given that we are able to find the approximate semantic vector from Equation (17) for a point, we should be able to find an approximate label with the mapping $\mathcal{L}$. Again, there is a multitude of ways to approach this secondary mapping, but as stated in Section 2 the semantic vectors in S are one-hot encodings for the labels in L, so the simplest way to accomplish this could be by finding the nearest neighbour $\mathbf{s}_c$ from the set of semantic vectors S and return the label associated with it $l_c \in L$.

5.2 Generic classification

The zero-shot capabilities arise from the intermediary mapping to the semantic space by computing the weighted average of the semantic vectors associated with each ellipsoid. However, if access to semantic vectors proves difficult they could be exchanged with vectors representing the centerpoints of the ellipsoids, thus giving a robust approximation for the centerpoint of the ellipsoid the point should belong to. That is if we define a set M as the set of centerpoints $\mathbf{m}$ of the ellipsoids (the means of the datapoints associated with each label) we can rewrite Equation (17) as:

$$\mathcal{S}(\mathbf{f}) = \frac{\sum_{\mathbf{m} \in M} K(d_{\mathbf{m}}(\mathbf{f}))\mathbf{m}}{\sum_{\mathbf{m} \in M} K(d_{\mathbf{m}}(\mathbf{f}))}$$

This can then be compared with existing centerpoints in the set M with a nearest neighbor search to reach an unambiguous classification.

Note that this way there is no possibility for classifying an unseen label, but this form of a generic classification can be useful in some applications or at least in benchmarking against other classification models.

5.3 Pseudocode

A simple pseudocode implementation tying all that has been covered is shown in Algorithm 1. Note that the operator $\circ$ is used to describe the element-wise product (or Hadamard product) between two matrices.

Algorithm 1: Basic classifier implementation

```
1  Initialise:
2  $F_{\text{train}}$; $F_{\text{test}}$, training and testing sets;
3  $L_0$; $L$, set of distinct labels available in training and set of all distinct labels respectively;
4  $S_0$; $S$, set of semantic vectors for labels in $L_0$ and set of semantic vectors for labels in $L$ respectively;
5  $E = \texttt{[]}$, empty list for the found ellipsoids;
6  $p = \texttt{[]}$, empty list of predictions;
7  for $l$ in $L_0$ do
8      $X, Y = \texttt{split}(F_{\text{train}}, l)$, split data points into set of points $X$ with label $l$ and set of points $Y$ with any other label $L_0 \setminus \{l\}$;
9      $\mathcal{E}_A(\mathbf{m})^{(l)} = \texttt{solve}(X, Y)$, solve the problem in Equation (14) to find the optimal ellipsoid;
10     $E.\texttt{append}(\mathcal{E}_A(\mathbf{m})^{(l)})$, save the found ellipsoid;
11 end
12 for $f$ in $F_{test}$ do
13     $d = \texttt{[]}$, empty list for the computed distances from the point $f$ to the ellipsoids;
14     for $\mathcal{E}_A(\mathbf{m})^{(l)}$ in $E$ do
15         $d.\texttt{append}(\texttt{kernelfunc}(\texttt{dist}(\mathcal{E}_A(\mathbf{m})^{(l)}, f)))$, compute the distance from the given ellipsoid and add the value returned by the kernel function of it to the list;
16     end
17     $s = \texttt{sum}(S_0 \circ \texttt{softmax}(d)))$, compute the weighted average of the semantic vectors corresponding with each label $l$ in $L_0$ as the sum of the element-wise products of the semantic vectors and their corresponding probability as given by the softmax function defined in Definition 12;
18     $p.\texttt{append}(\texttt{nn}(s, S, L))$, find the nearest neighbor for the approximate point $s$ from $S$ and add the corresponding label $l \in L$ to the predictions;
19 end
20 return p;
```

5.4 Limitations

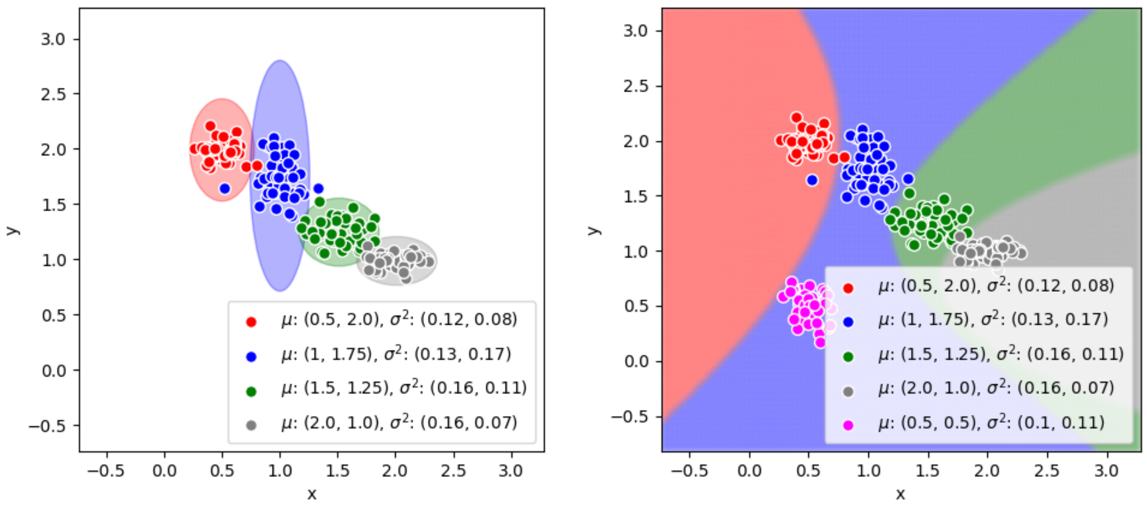

Figure 1: Left: Visualization of the ellipsoids found in the training. Right: Visualization of the decision boundaries for the labels used in training and an outlying unseen label

Even before testing the algorithm we can note some theoretical limitations it will have, especially considering the zero-shot capabilities of it. The largest issue comes when trying to zero-shot predict some outlier. Consider for example the ellipses found in Figure 1. If we had as training data the labels in the left figure and had to predict a point that belongs to a label that is a clear outlier the model clearly fails as seen in the right figure. This is because the approximated semantic vector would be the average of the semantic vectors associated with labels used in training and thus should be far closer to them than the outlier (assuming a good choice of semantic space).

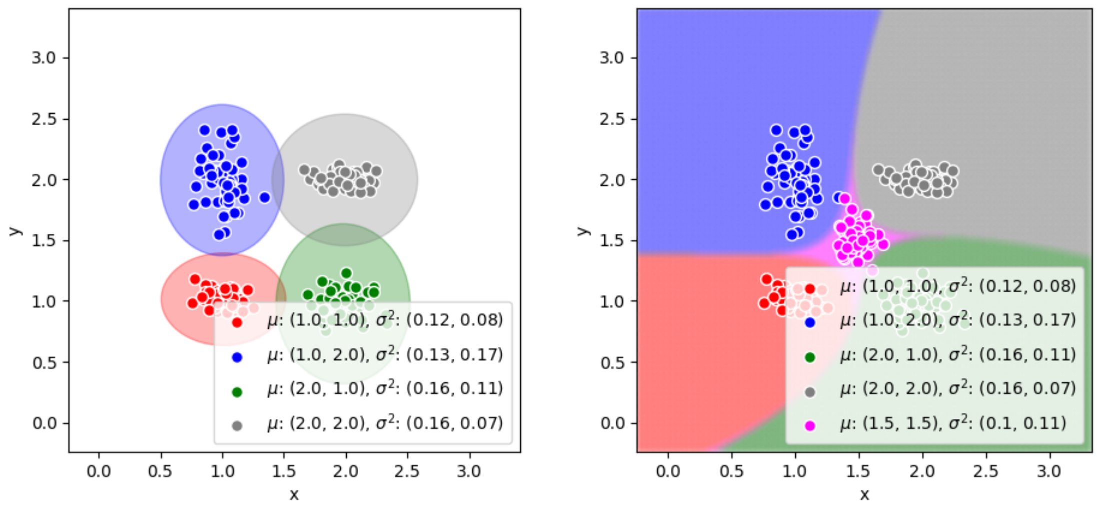

Figure 2: Left: Visualization of the ellipsoids found in the training. Right: Visualization of the decision boundaries for the labels used in training and a well situated unseen label

An example of the other extreme, where a good prediction would be almost guaranteed, can be seen in Figure 2. Here if the training data consisted of labels associated with the ellipsoids seen in the left figure and we had to predict a point that falls within the unseen label in the center of them we should have a good chance for succeeding as each of the training ellipses get roughly an equal weight placing the approximated semantic vector close to the correct region. This can be seen in the right figure of Figure 2. In general then we would like to have as many labels as possible in the training data as then the probability that an unseen label is an outlier decreases and we can expect better zero-shot performance.

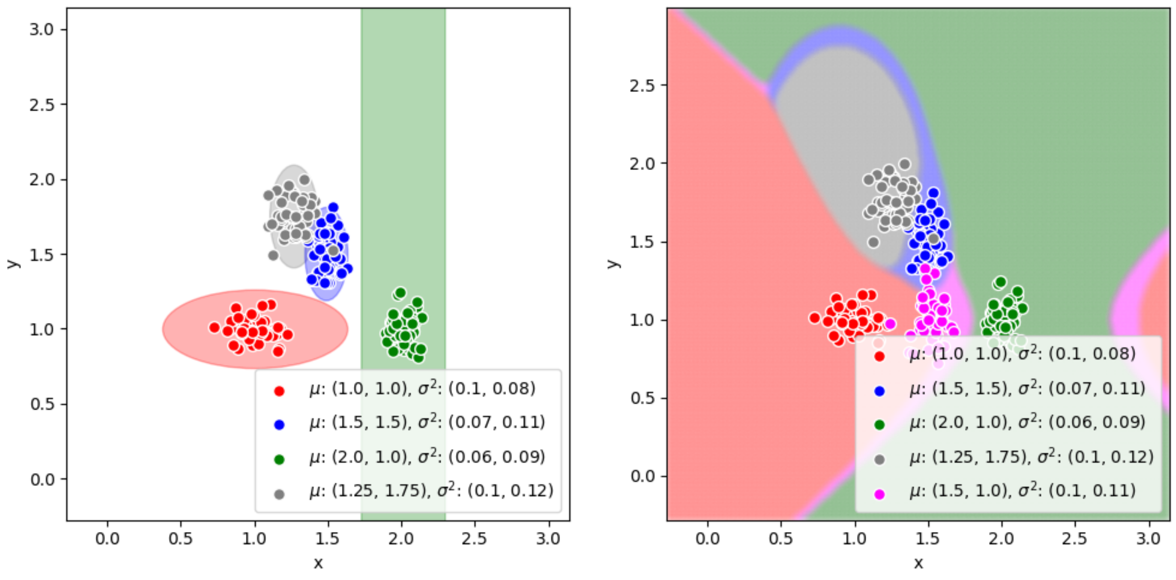

Figure 3: Left: Visualization of the ellipsoids found in the training with an ellipsoid with an unbounded optimization problem. Right: Visualization of the decision boundary for the labels used in training and an unseen label

Another issue might arise when the points in set X, which the ellipsoid is trying to encompass are not surrounded along each principle axis by points in set Y. In this case the surface that should maximize the distance to the closest points in both sets X and Y does not exist and the optimization problem ends up maximizing the distance from the closest point in set X without a bound. An example of this is visualized in Figure 3.

6 Implementation

6.1 Full model

While the problem definition in Equation (14) is correct it can be slightly improved. Firstly, not all solvers support matrix operations so in general it would be better to write everything open as a sum. Secondly, the properties of the characteristic matrix are not utilized. Namely, we are multiplying a symmetric matrix from both sides with the same vector, which results in a quadratic term. That is if we have some arbitrary vector $\mathbf{v} \in \mathbb{R}^n$ and a symmetric matrix $A \in \mathbb{R}^{n \times n}$ we find that:

$$\mathbf{v}^T A \mathbf{v} = \sum_{i=1}^{n} (a_{ii} v_i^2) + 2 \sum_{i=2}^{n} \sum_{j=1}^{i-1} (a_{ij} v_i v_j)$$

where a_{ij} is the element in row i and column j in the matrix A. This also means we only need to have decision variables for the elements on the diagonal and the lower triangular of the matrix, meaning in total the number of decision variables is:

$$\text{nvars}(A) = n\frac{n+1}{2}$$

In our interpretation, where the inverse of the characteristic matrix of the ellipsoid is an approximation for the covariance matrix, the elements on the diagonal represent the variances of the features and thus they must be non-negative while no such restriction holds for the elements outside the diagonal. This gives us a improved problem formulation of form:

$$\begin{aligned}
\min_{A} \quad & \sum_{k=1}^{|X|} u_k + \sum_{k=1}^{|Y|} v_k \\
\text{s.t.:} \quad & u_k - \sum_{i=1}^{n}(a_{ii}(x_{ki} - m_i)_i^2) - 2\sum_{i=2}^{n}\sum_{j=1}^{i-1}(a_{ij}(x_{ki} - m_i)(x_{kj} - m_j)) \geq 0, && \forall k \in \{1, ..., |X|\} \\
& v_k + \sum_{i=1}^{n}(a_{ii}(y_{ki} - m_i)_i^2) + 2\sum_{i=2}^{n}\sum_{j=1}^{i-1}(a_{ij}(y_{ki} - m_i)(y_{kj} - m_j)) - 2 \geq 0, && \forall k \in \{1, ..., |Y|\} \\
& u_k \geq 0, && \forall k \in \{1, ..., |X|\} \\
& v_k \geq 0, && \forall k \in \{1, ..., |Y|\} \\
& a_{ii} \geq 0, && \forall i \in \{1, ..., n\}
\end{aligned} \tag{19}$$

Even with these improvements the main issue of the model is that the number of decision variables needed grows at a rate $\mathcal{O}(n^2)$. Thus, the problem becomes very challenging and time-consuming to solve already with datapoints with a few thousand features. Some simplifications are in order.

6.2 Independent model

If we make the assumption that the features are independent of each other the only thing we would need to model are the variances of each feature, which are located on the diagonal of the characteristic matrix. With this assumption the number of needed decision variables grows at a rate $\mathcal{O}(n)$, where n is the number of features, greatly simplifying the problem.

With this simplification the problem formulation would be:

$$
\begin{aligned}
\min_{\mathbf{a}} \quad & \sum_{i=1}^{|X|} u_i + \sum_{i=1}^{|Y|} v_i \\
\text{s.t.:} \quad & u_i - (\mathbf{x}_i - \mathbf{m})^T(\mathbf{a} \circ (\mathbf{x}_i - \mathbf{m})) \geq 0, && \forall i \in \{1, ..., |X|\} \\
& v_i + (\mathbf{y}_i - \mathbf{m})^T(\mathbf{a} \circ (\mathbf{y}_i - \mathbf{m})) - 2 \geq 0, && \forall i \in \{1, ..., |Y|\} \\
& u_i \geq 0, && \forall i \in \{1, ..., |X|\} \\
& v_i \geq 0, && \forall i \in \{1, ..., |Y|\} \\
& \mathbf{a} \geq \mathbf{0}
\end{aligned}
\tag{20}
$$

where $\text{diag}(A) = \mathbf{a}$ and $\circ$ is again used to denote the element-wise product (Hadamard product) between two matrices.

However, this model runs into issues with more complicated data as understandably the assumption of independence is rarely accurate.

6.3 Banded model

What we would like to find is some middle ground between the assumptions discussed in Sections 6.1 and 6.2 that would tackle their drawbacks. And indeed the middle ground is an apt term to use as a good choice for such a model would be a "banded" one, i.e., one in which the characteristic matrix is banded. There are a few benefits that such a model would have.

Firstly, the banded model can be chosen to be such that it takes advantage of some structure that we can assume to exist in the data. Consider for example that we have some image data. Images have a clear two-dimensional structure with the pixel values and usually for humans the images are only understandable in their original form. However, if we were to pass to the full model images that are all shuffled, but shuffled in the same way for each image, the model would work equally well as in the unshuffled case, while for humans the task becomes impossible. This is because swapping two pixel values in all images corresponds with swapping the respective columns and rows in the characteristic matrix. It is clear then that using the full model means we end up solving a harder problem than is necessary. Thus, it might be worthwhile to only consider some subset of possible features when computing the dependencies, for example we could only look at the eight neighbouring pixels and find the covariances for them as most likely the dependencies are greater in nearby pixels than far away ones.

Secondly, only computing the covariances for some subset of features means the total number of parameters needed is significantly less than with the full model. Unfortunately, as the shape of the band is dependent on the structure we are looking for we don't have a closed form equation to compute the precise number of varibles needed, but it needs to be done on case by case basis.

For simpler notation we can define a function:

$$g(A, X, \mathbf{m}, k, j) = \sum_{i=j}^{n} (a_{i(i-j)}(x_{ki} - m_i)(x_{k(i-j)} - m_{(i-j)})$$

where $A \in \mathbb{R}^{n \times n}$, $X \in \mathbb{R}^{|X| \times n}$, $\mathbf{m} \in \mathbb{R}^n$ and $k, j \in \mathbb{N}$. The optimization problem for banded model would then be:

$$\begin{aligned}
\min_{A} \quad & \sum_{k=1}^{|X|} u_k + \sum_{k=1}^{|Y|} v_k \\
\text{s.t.:} \quad & u_k - \sum_{i=1}^{n} (a_{ii}(x_{ki} - m_i)_i^2) - 2 \sum_{\Delta b \in B} g(A, X, \mathbf{m}, k, \Delta b) \geq 0, && \forall k \in \{1, ..., |X|\} \\
& v_k + \sum_{i=1}^{n} (a_{ii}(y_{ki} - m_i)_i^2) + 2 \sum_{\Delta b \in B} g(A, Y, \mathbf{m}, k, \Delta b) - 2 \geq 0, && \forall k \in \{1, ..., |Y|\} \\
& u_k \geq 0, && \forall k \in \{1, ..., |X|\} \\
& v_k \geq 0, && \forall k \in \{1, ..., |Y|\} \\
& a_{ii} \geq 0, && \forall i \in \{1, ..., n\}
\end{aligned} \tag{21}$$

where B is the set of deviations from the main diagonal corresponding to the shape of the band around the feature. Note that as the characteristic matrix A is symmetric we only need to define the lower or upper triangular form of the band in B.

6.4 Decomposition

Sometimes the dataset for a given optimization problem is still far too large for any of the above discussed models to be trained in reasonable amount of time. In these types of large scale problems it is usually necessary to try to break the problem into smaller parts that could be solved in shorter time and possibly even in parallel. But how could one decompose the problem at hand into more reasonable chunks?

Again let us consider that we have a n-dimensional random vector $\mathbf{X} \in \mathbb{R}^n$. Consider that the features in $\mathbf{X}$ are t time-varied measurements from k sensors (so that $n = k \cdot t$). Then the random vector can be written as $\mathbf{X} = [\mathbf{X}_1^T \quad \mathbf{X}_2^T \quad ... \quad \mathbf{X}_k^T]^T$, where $\mathbf{X}_i$ is the random vector of measurements from sensor i. Using the block matrix notation we can write the covariance matrix of $\mathbf{X}$ as:

$$K_{\mathbf{XX}} = \begin{bmatrix} K_{\mathbf{X}_1\mathbf{X}_1} & K_{\mathbf{X}_1\mathbf{X}_2} & \cdots & K_{\mathbf{X}_1\mathbf{X}_k} \\ K_{\mathbf{X}_2\mathbf{X}_1} & K_{\mathbf{X}_2\mathbf{X}_2} & \cdots & K_{\mathbf{X}_2\mathbf{X}_k} \\ \vdots & \vdots & \ddots & \vdots \\ K_{\mathbf{X}_k\mathbf{X}_1} & K_{\mathbf{X}_k\mathbf{X}_2} & \cdots & K_{\mathbf{X}_k\mathbf{X}_k} \end{bmatrix} \tag{22}$$

There is no good interpretation in our model for covariance matrices between two different random vectors, but matrices $K_{\mathbf{X}_i\mathbf{X}_i}$ are equally valid for solving as the full problem. Thus, we could do the optimization for them independently with, for example the banded model and only afterwards combine them back together into the full characteristic matrix.

However, there are two things we should note before this. Firstly, the matrix shown in Equation (22) only holds for covariance matrices, yet the ellipsoid is defined with the inverse of such. But as we are only considering a block diagonal it can be shown that:

$$A = K^{-1}_{\mathbf{X}_i\mathbf{X}_i} = \begin{bmatrix} K_{\mathbf{X}_1\mathbf{X}_1} & & \\ & \ddots & \\ & & K_{\mathbf{X}_k\mathbf{X}_k} \end{bmatrix}^{-1} = \begin{bmatrix} K^{-1}_{\mathbf{X}_1\mathbf{X}_1} & & \\ & \ddots & \\ & & K^{-1}_{\mathbf{X}_k\mathbf{X}_k} \end{bmatrix} \tag{23}$$

This is proven in the following theorem:

Theorem 13. *The inverse of a block diagonal matrix is equivalent to the matrix with the inverses of the block matrices on the diagonal.*

Proof. Assume we have a given block diagonal matrix $A = \text{diag}(A_1, A_2, ..., A_k) \in \mathbb{R}^{n\times n}$. Furthermore, assume that the block matrices $A_i \in \mathbb{R}^{p\times p}$ are invertible. Define another block diagonal matrix as $B = \text{diag}(A_1^{-1}, A_2^{-1}, ..., A_k^{-1}) \in \mathbb{R}^{n\times n}$. Then by rules of block matrix multiplication:

$$\begin{aligned} AB &= \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_k \end{bmatrix} \begin{bmatrix} A_1^{-1} & & \\ & \ddots & \\ & & A_k^{-1} \end{bmatrix} \\ &= \begin{bmatrix} A_1A_1^{-1} & & \\ & \ddots & \\ & & A_kA_k^{-1} \end{bmatrix} \\ &= \begin{bmatrix} I & & \\ & \ddots & \\ & & I \end{bmatrix} = I \end{aligned}$$

Thus, must hold that $\text{diag}(A_1, A_2, ..., A_k)^{-1} = \text{diag}(A_1^{-1}, A_2^{-1}, ..., A_k^{-1})$. ◻

Secondly, we should note that using this decomposition leads to “gaps” where the matrix transitions from one block matrix to the next. That is there can be elements of the matrix that are part of the band of interest, but which does not belong to any of the block diagonal matrices. This can be visualized for example with a case where the matrix is divided into 2×2 blocks as in Equation (24). In Equation (24) the gap elements are colored red.

$$K_{\mathbf{X}\mathbf{X}} = \begin{bmatrix} K_{\mathbf{X}_1\mathbf{X}_1} & K_{\mathbf{X}_1\mathbf{X}_2} \\ K_{\mathbf{X}_2\mathbf{X}_1} & K_{\mathbf{X}_2\mathbf{X}_2} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \times & \times & \dots & 0 \\ \times & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \times \\ 0 & \dots & \times & \times \end{bmatrix} & \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \times & \dots & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & \dots & \times \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 \end{bmatrix} & \begin{bmatrix} \times & \times & \dots & 0 \\ \times & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \times \\ 0 & \dots & \times & \times \end{bmatrix} \end{bmatrix} \tag{24}$$

This gap is indeed a problem and requires thought from the user. However in our example, where we have m sensors this gap coincides with the dependencies of the last few (depending on the choice of B) measurements of one sensor with the first few measurements from the next sensor and thus should not be all that interesting and losing that information is not detrimental. If we had chosen the example to be something where the gap does not have a justifiable interpretation, we could try to fix it for example by doing an additional optimization for the whole model knowing the gapped characteristic matrix, but having as parameter only the dependencies coinciding with the gaps. However, this approach is not discussed further here.

It is good to consider what this decomposition means geometrically as it might not be intuitively clear. Technically, we end up finding ellipsoids that enclose the given datapoints in a robust fashion in a subspace of the feature space. When these ellipsoids are found we combine them together to form an ellipsoid that encompasses the datapoints in the full feature space. But why should this work? As mentioned in Theorem 7 the eigenvectors and eigenvalues of the characteristic matrix define the ellipsoid and as it happens when we combine the block matrices on to the diagonal the eigenvectors and eigenvalues of the full characteristic matrix will be the union of the eigenvectors (padded with zeros to match the dimension) and eigenvalues of the individual block matrices as proven by following theorem:

Theorem 14. *The union of the eigenvectors (with zeros added to match the dimension) and eigenvalues of the block diagonal matrices is equal to the set of eigenvectors and eigenvalues of the full matrix.*

Proof. Assume we have a block diagonal matrix $A = \text{diag}(A_1, A_2, ..., A_k) \in \mathbb{R}^{n\times n}$. Furthermore, assume that for a given block matrix $A_i \in \mathbb{R}^{p\times p}$ we know the j-th eigenvalue λ_{ij} and the corresponding eigenvector $\mathbf{v}_{ij}$ that satisfy the eigenvalue equation:

$$A_i\mathbf{v}_{ij} = \lambda_{ij}\mathbf{v}_{ij}$$

Consider a vector $\mathbf{w} = [\mathbf{w}_1^T \quad \mathbf{w}_2^T \quad ... \quad \mathbf{w}_k^T]^T$, where $\mathbf{w}_{i_0} \in \mathbb{R}^p$ for all $i_0 \in \{1, ..., k\}$. Define $\mathbf{w}$ so that $\mathbf{w}_i = \mathbf{v}_{ij}$ and $\mathbf{w}_{i_0} = \mathbf{0}$ for all $i_0 \in \{1, ..., k\} \setminus \{i\}$. Then by block matrix multiplication we find:

$$A\mathbf{w} = \begin{bmatrix} A_1 & & & & \\ & \ddots & & & \\ & & A_i & & \\ & & & \ddots & \\ & & & & A_k \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{v}_{ij} \\ \vdots \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ A_i\mathbf{v}_{ij} \\ \vdots \\ \mathbf{0} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{v}_{ij} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

As the choice of block matrix and eigenvector of said matrix was arbitrary, then the eigenvalues of the block matrices are also the eigenvalues of the full matrix and the corresponding eigenvectors are the eigenvectors of the block matrix buffered with zeros. Note that as a $n \times n$ matrix can have most n eigenvalues, a block diagonal matrix cannot have any other eigenvalues or eigenvectors than those of the block matrices since they already constitute the $p \cdot k = n$ eigenvalues and eigenvectors. □

6.5 Generic performance optimization

Additional consideration can be given to some universal factors that may affect the performance. One obvious thing to note is the clear disparity between the number of datapoints in set X as opposed to set Y as clearly when there are multiple labels in the classification problem $|Y| \gg |X|$. As in the optimization problem we have decision variables for each datapoint the size of Y can end up having significant impact on the problem as a whole. Thus, it might be worthwhile to only consider a subset Y_0 of Y that is randomly sampled.

Additionally, in some cases due to the great size difference between sets X and Y it might be that the solver ends up focusing more on minimizing the sum $\sum_{k=1}^{|Y|} v_k$ than $\sum_{k=1}^{|X|} u_k$ as in most cases the found values $u_k \ \forall \ k \in 1, ..., |X|$ and $v_k \ \forall \ k \in 1, ..., |Y|$ end up with some very small, but non-zero value. This could be combatted by adding a multiplier ω to the sum $\sum_{k=1}^{|X|} u_k$ to balance this disparity. Trivial choice for ω would be $\omega = |Y|/|X|$, but other could be tested depending on the task at hand.

7 Results

We will test the outlined algorithm in two cases. Firstly, we will test it with randomized points and mainly study how the choice of semantic vectors and the number of labels left out of the training data affect the prediction accuracy. We compare the generic algorithm with the k-nearest neighbour (KNN) algorithm [29]. The k-nearest neighbour algorithm in its simplicity takes the k nearest training examples to a given point and classifies the point under the most common label in said examples. Thus, it works great as a comparison since it similarly depends on the points for a given label to be nearby in the feature space and can work with relatively small number of training points. Secondly, we will test the generic algorithm with a real (although simple) dataset of handwritten digits and similarly compare the performance to the k-nearest neighbour algorithm.

The source code used in these experiments is open-source and can be found on GitHub at: `https://github.com/krantamaki/Robust_Ellipsoid_Classifier`

7.1 Randomized points

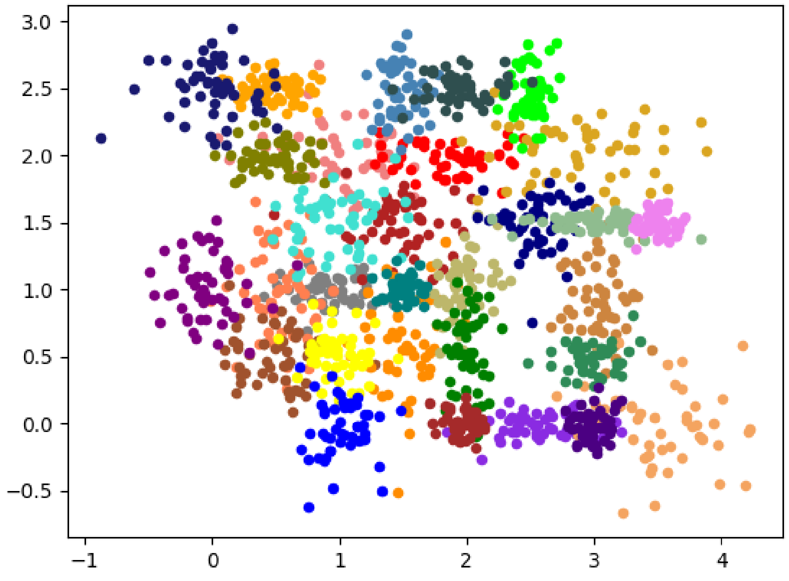

Figure 4: 50 randomly generated points for each label outlined in Table A1

Our randomized data consists of random datapoints each with features acquired from a variety of normal distributions. The sets of means and variances can be found from Appendix A in Table A1. There are 30 of such sets meaning we have 30 labels to use. For visualization purposes the sets are chosen to define points in 2-dimensional space. In this case the features are also clearly independent and thus we can utilize the implementation with the independence assumption.

We generated 25 datapoints for each label for training and similarly 25 datapoints per label for testing. One example of the datapoints can be seen in Figure 4. It is clear that the classification problem is not exactly trivial. Furthermore, as the labels to be left out of training are chosen randomly each time, results from ten random choices of left out labels are averaged together to form the final prediction accuracies.

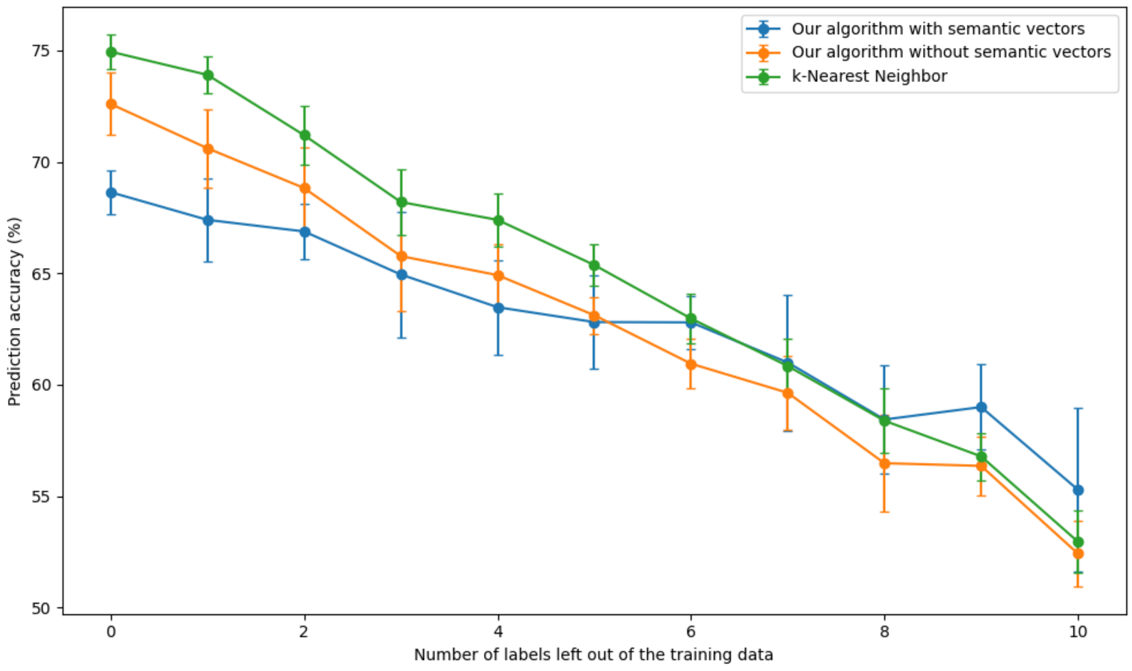

Figure 5: Means and standard deviations of the results from ten random choices of left out labels with semantic vectors defined by a nonlinear mapping compared to k-nearest neighbors algorithm

With randomized data an obvious question becomes what to use as the semantic vectors. In this case we have two different mappings for the means that map to a higher dimension. The choice to use a higher dimension is arbitrary. Commonly, the semantic space is of lower dimension, but as having an unambigous one dimensional semantic space would most likely require quite complex hash functions, it is easier to make the mapping to a higher dimension. Firstly, we use an arbitrarily chosen nonlinear map to find the semantic vectors. The results with this nonlinear mapping are shown in Figure 5. It is clear that the performance is not all that great. In general it does not seem that the semantic vectors provide much extra information to the classifier as the generic version shows similar performance even with multiple left out labels and neither equals the predictive accuracy of the k-nearest neighbor algorithm, until a very large number of left out labels. Most likely the cause of this are distorted relative distance caused by the nonlinear nature of the mapping.

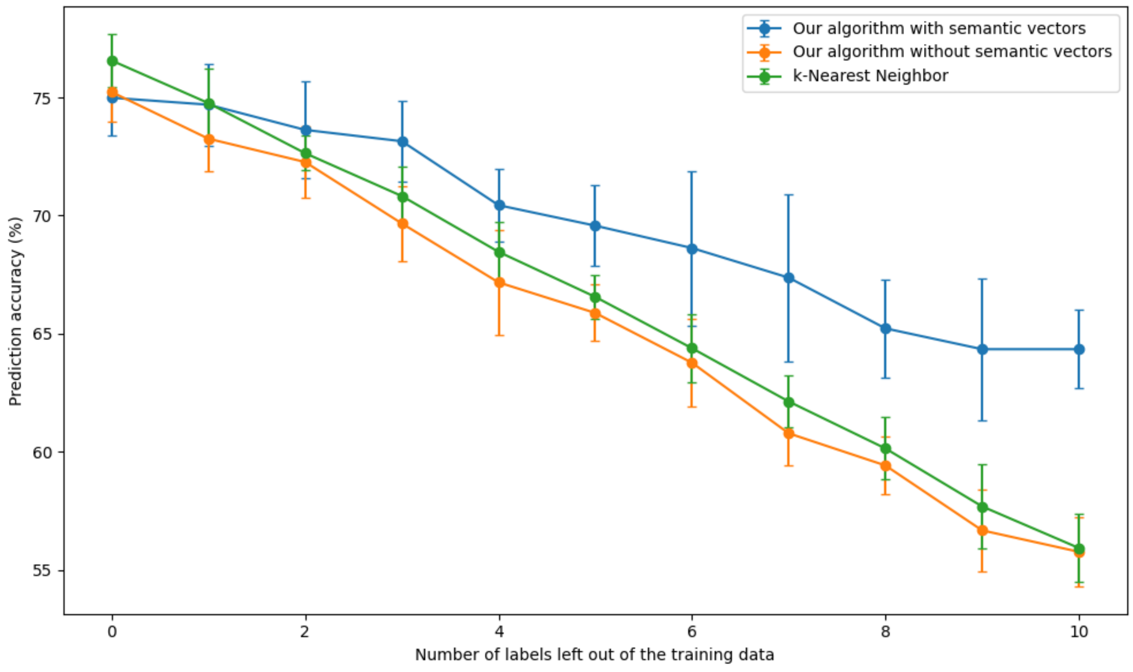

Figure 6: Means and standard deviations of the results from ten random choices of left out labels with semantic vectors defined by a linear mapping compared to k-nearest neighbors algorithm

Secondly, we use an arbitrarily chosen mapping of linear form, which should better maintain the relative locations in the semantic space. The results with this linear mapping are shown in Figure 6. This mapping performs understandably much better. Indeed, it out-competes the generic classifier in almost every case and k-nearest neighbor algorithm in most cases. The most impressive part however is how with the semantic information the algorithm is at times able to maintain almost constant prediction accuracy as the number of left out labels grows. In contrast for both k-nearest neighbor algorithm and the generic classifier the prediction accuracy decreases in a very linear fashion.

Of course the choice of a linear map can be considered a bit questionable as by the nature of linearity the mapping cannot be surjective. Thus, the semantic space ends up being only a stretched and rotated version of the feature space, meaning that for the most part the result is the same as if one used just the means as the semantic vectors. This shows a crucial limitations for the algorithm in assuming that the relative distances between points in the semantic space and feature space are close to identical. But one could perhaps justify this if the source of the semantic and feature space is the same in some fundamental way. Whether this could be accomplished in practice is a completely separate issue and requires thought from the user.

7.2 MNIST dataset

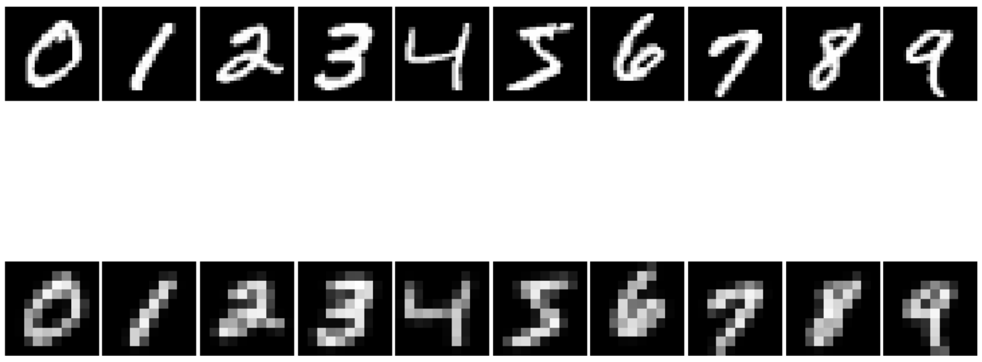

Figure 7: Examples of MNIST dataset digits in the full 28×28 pixels and reduced to 10×10 pixels

In the second experiment we use the MNIST (Modified National Institute of Standards and Technology) dataset [30]. This dataset consists of 60 000 training images and 10 000 testing images of handwritten digits, which are converted into black and white and normalized to fit into 28×28 pixels. This is a good choice for testing as it is very clear how each handwritten digit could follow some unknown probability distribution.

We will reduce the image resolution from the full 28×28 pixels to 10×10 pixels to reduce the computational burden. Examples of such reduced digits can be seen in Figure 7. We will use the banded model from Section 6.3. The band will be defined as a 3×3 grid of pixels from around the pixel of interest. We will train the model on 400 datapoints per label and similarly use 200 datapoints per label for testing. We will use the general classifier only as there is not enough distinct labels for proper zero-shot predictions given the limitations discussed in Section 5.4.

The results can be found from Table 1.

Table 1: Prediction accuracies for our banded model algorithm and KNN for 5 different samples with 400 training points and 200 testing points per label

	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Our Algorithm	0.871	0.867	0.881	0.869	0.886
KNN	0.924	0.915	0.9155	0.9145	0.9285

The results seem very convincing and are clearly far greater than what would be obtained by pure chance. And while our algorithm does not quite reach the performance of the KNN algorithm, the prediction accuracy is still decent considering the limited amount of data used. Perhaps if the width of the band was increased the results could be improved, but these results are more than satisfactory to verify the functioning of our classifier.

8 Conclusions

In this thesis we have covered the basics of zero-shot learning and outlined a novel model for variety of zero-shot learning tasks. In the results of Section 7 we find that the model can compete with already well-established algorithms in different classification tasks. But one could argue that these tasks were very simple compared to many seen in industry. And this is true. The outlined model would be hopeless in a classification task like the *ImageNet Large Scale Visual Recognition Challenge* discussed in Section 1. Indeed same holds in any task where the probability distributions are not relatively simple, which most likely includes a vast majority of real world applications. This is something that could be overcome though, by not training our model directly with the raw datapoints, but training for example an autoencoder or some other more flexible model for finding a denser representation for the data. The hope would be that in this denser representation the datapoints are more neatly clustered and would end up following some simple distribution that our algorithm would find. Additional benefit of an approach like this would be that the feature space passed to our algorithm is of lower dimension than the original and thus the optimization problem would be significantly easier.

Second challenge for our model, and for any zero-shot learning model for that matter, is the access to a valid semantic space. In Section 2 we discuss the *word2vec* vectors, which are a common choice for semantic vectors, but depend heavily on the corpus from which they are generated. Our model assumes that the relative locations of the labels in the feature space and in the semantic space are comparable, but this is not necessarily the case. A good example would be the word2vec vector for the word “lion” in the Finnish word2vec model, which happens to be closer to ice hockey related vocabulary than other animals as one might assume. This is of course a result of the fact that the Finnish men’s national hockey team is commonly referred to as the “lions” (“leijonat”) in news articles and other text sources used in the generation of these word2vec vectors.

More generally it might appear that the zero-shot framework as a whole is more restricting than necessary. Indeed it is very rarely the case that no data would exist for some labels of interest, even if generating the data is very expensive. It is a lot more common that the amount of data is uneven between the labels. These cases are studied in the field of *few-shot learning* of which zero-shot learning can be considered a subfield. And indeed our algorithm can easily be adapted for few-shot learning tasks. In the training phase one could train the ellipsoids for only the subset of the labels, for which there exists the most data, but pass as “semantic vectors” the means for all labels, including the ones not used in training. Then as in the case of zero-shot learning our model could make predictions for labels with the limited amount of data.

This all shows the immense flexibility, but also challenge, which comes with zero-shot learning and is not limited to the model proposed in this thesis. Indeed for these exact reasons zero-shot learning is a very exiting emerging field, that hopefully will in time open new avenues for machine learning and make it evermore widely adopted in a variety of applications.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, pp. 84 – 90, 2012.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[4] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-shot text-to-image generation,” *International Conference on Machine Learning*, pp. 8821–8831, 2021.

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[6] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[10] S. L. Kivisaari, M. van Vliet, A. Hultén, T. Lindh-Knuutila, A. Faisal, and R. Salmelin, “Reconstructing meaning from bits of information,” *Nature communications*, vol. 10, no. 1, p. 927, 2019.

[11] G. Sudre, D. Pomerleau, M. Palatucci, L. Wehbe, A. Fyshe, R. Salmelin, and T. Mitchell, “Tracking neural coding of perceptual and semantic features of concrete nouns,” *NeuroImage*, vol. 62, no. 1, pp. 451–463, 2012.

[12] T. M. Mitchell, S. V. Shinkareva, A. Carlson, K.-M. Chang, V. L. Malave, R. A. Mason, and M. A. Just, “Predicting human brain activity associated with the meanings of nouns,” *Science*, vol. 320, no. 5880, pp. 1191–1195, 2008.

[13] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, “Zero-shot learning with semantic output codes,” *Advances in neural information processing systems*, vol. 22, 2009.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Advances in neural information processing systems*, vol. 26, 2013.

[16] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, “Zero-shot learning – a comprehensive evaluation of the good, the bad and the ugly,” *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[17] Y. Xian, B. Schiele, and Z. Akata, “Zero-shot learning-the good, the bad and the ugly,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4582–4591, 2017.

[18] E. Kodirov, T. Xiang, and S. Gong, “Semantic autoencoder for zero-shot learning,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3174–3183, 2017.

[19] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” *International conference on machine learning*, 2021.

[20] J. A. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural processing letters*, vol. 9, pp. 293–300, 1999.

[21] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, “Support vector regression machines,” *Advances in neural information processing systems*, vol. 9, 1996.

[22] P. K. Shivaswamy and T. Jebara, “Ellipsoidal machines,” *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, pp. 484–491, 2007.

[23] A. Astorino, A. Frangioni, E. Gorgone, and B. Manca, “Ellipsoidal classification via semidefinite programming,” *Operations Research Letters*, vol. 51, no. 2, pp. 197–203, 2023.

[24] M. Rosenblatt, “Remarks on some nonparametric estimates of a density function,” *The annals of mathematical statistics*, pp. 832–837, 1956.

[25] E. Parzen, “On estimation of a probability density function and mode,” *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[26] E. A. Nadaraya, “On estimating regression,” *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[27] G. S. Watson, “Smooth regression analysis,” *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

[28] M. P. Chandra *et al.*, “On the generalised distance in statistics,” *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.

[29] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[30] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

A Additional material

Table A1: Means and standard deviations of the used labels

μ_x	μ_y	σ_x^2	σ_y^2
1.00	1.00	0.20	0.20
1.00	2.00	0.30	0.30
1.50	1.50	0.30	0.30
2.00	2.00	0.30	0.30
0.50	1.00	0.20	0.20
0.50	0.50	0.20	0.20
3.00	1.00	0.20	0.20
1.50	0.50	0.20	0.20
3.00	2.00	0.40	0.40
2.00	1.00	0.15	0.15
0.50	2.00	0.20	0.20
1.00	0.50	0.15	0.15
2.00	0.50	0.10	0.10
2.50	2.50	0.10	0.10
1.00	1.50	0.30	0.30
1.50	1.00	0.10	0.10
1.50	2.50	0.15	0.15
2.50	1.50	0.20	0.20
1.00	0.00	0.15	0.15
2.50	0.00	0.30	0.30
0.00	1.00	0.20	0.20
2.00	0.00	0.10	0.10
3.50	0.00	0.30	0.30
0.50	2.50	0.20	0.20
3.00	1.50	0.30	0.30
3.00	0.50	0.20	0.20
2.00	2.50	0.20	0.20
0.00	2.50	0.30	0.30
3.00	0.00	0.10	0.10
3.50	1.50	0.10	0.10