Aalto University
School of Science

Zero-shot classifier based on robust ellipsoid optimization

Kasper Rantamäki

July 21, 2023

0 Overview

1

Introduction

1 Machine learning classifiers

- Classifiers (generally) find a mapping:

$$\mathcal{F} : F^d \to L \qquad (f : X^d \to Y)$$

1 Machine learning classifiers

- Classifiers (generally) find a mapping:

$$\mathcal{F} : F^d \to L \qquad (f : X^d \to Y)$$

- i.e. divide the feature space into regions for each label with *decision boundaries*

1 Machine learning classifiers

- Classifiers (generally) find a mapping:

$$\mathcal{F} : F^d \to L \qquad (f : X^d \to Y)$$

- i.e. divide the feature space into regions for each label with *decision boundaries*
- Optimal mapping generally divides the whole space between the labels seen in training

1 Machine learning classifiers

- Classifiers (generally) find a mapping:

$$\mathcal{F} : F^d \to L \qquad (f : X^d \to Y)$$

- i.e. divide the feature space into regions for each label with *decision boundaries*
- Optimal mapping generally divides the whole space between the labels seen in training
- Thus, not well suited for classifying labels not seen in training

1 Zero-shot classifiers

- (Palatucci et al. 2009)[1] propose a *semantic output code classifier* as a composition of two different mapping functions:

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

1 Zero-shot classifiers

- (Palatucci et al. 2009)[1] propose a *semantic output code classifier* as a composition of two different mapping functions:

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

- S^p is a semantic space containing *one-hot encodings* of labels

1 Zero-shot classifiers

- (Palatucci et al. 2009)[1] propose a *semantic output code classifier* as a composition of two different mapping functions:

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

- S^p is a semantic space containing *one-hot encodings* of labels
- One-hot encodings make the mapping $\mathcal{L}$ trivial

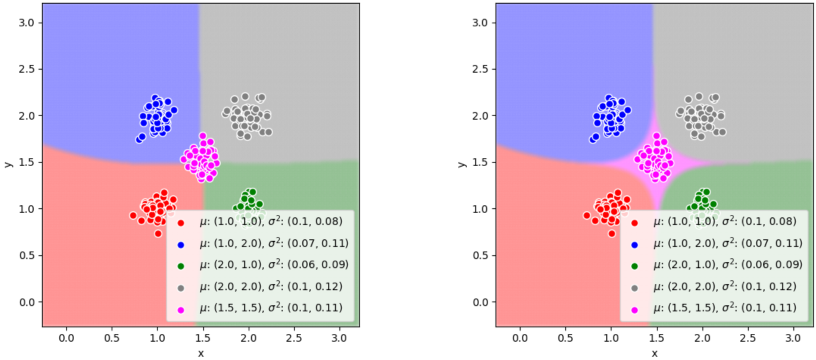
1 Zero-shot classifiers

- (Palatucci et al. 2009)[1] propose a *semantic output code classifier* as a composition of two different mapping functions:

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

- S^p is a semantic space containing *one-hot encodings* of labels
- One-hot encodings make the mapping $\mathcal{L}$ trivial
- Real effort is in finding the mapping $\mathcal{S}$

1 Visual comparison



2

Mathematical formulation

2 The “Goal”

- Find ellipsoids that best “fit” the datapoints of a given label

2 The "Goal"

- Find ellipsoids that best "fit" the datapoints of a given label
- Use metric induced by the ellipsoids in kernel regression as mapping $\mathcal{S}$

2 The “Goal”

- Find ellipsoids that best “fit” the datapoints of a given label
- Use metric induced by the ellipsoids in kernel regression as mapping $\mathcal{S}$
- Find the nearest neighbor for the approximation given by kernel regression from the set of semantic vectors

2 The "Goal"

- Find ellipsoids that best "fit" the datapoints of a given label
- Use metric induced by the ellipsoids in kernel regression as mapping $\mathcal{S}$
- Find the nearest neighbor for the approximation given by kernel regression from the set of semantic vectors
- Return the label of the nearest semantic vector as mapping $\mathcal{L}$

2 Probabilistic interpretation

- Consider a set of d-dimensional points $\mathbf{x} \in X$

2 Probabilistic interpretation

- Consider a set of d-dimensional points $\mathbf{x} \in X$
- Points can be modeled with a random vector:

$$\mathbf{X} = [X_1 \quad X_2 \quad \dots \quad X_d]^T$$

2 Probabilistic interpretation

- Consider a set of d-dimensional points $\mathbf{x} \in X$
- Points can be modeled with a random vector:

$$\mathbf{X} = [X_1 \quad X_2 \quad \dots \quad X_d]^T$$

- Measure of location modeled by the (arithmetic) mean $\mathbf{m}$

2 Probabilistic interpretation

- Consider a set of d-dimensional points $\mathbf{x} \in X$
- Points can be modeled with a random vector:

$$\mathbf{X} = [X_1 \quad X_2 \quad \dots \quad X_d]^T$$

- Measure of location modeled by the (arithmetic) mean $\mathbf{m}$
- Measure of scatter modeled by the covariance matrix $K_{\mathbf{XX}}$

2 Ellipsoidal approximation

- By properties of covariance $K_{\mathbf{XX}}$ is (symmetric) positive semi-definite

2 Ellipsoidal approximation

- By properties of covariance $K_{\mathbf{XX}}$ is (symmetric) positive semi-definite
- We assume $K_{\mathbf{XX}}$ is (symmetric) positive definite

2 Ellipsoidal approximation

- By properties of covariance $K_{\mathbf{XX}}$ is (symmetric) positive semi-definite
- We assume $K_{\mathbf{XX}}$ is (symmetric) positive definite
- Then $\mathbf{m}$ and $K_{\mathbf{XX}}$ can be used to define an ellipsoid as the set of points:

$$\mathcal{E}_A(\mathbf{m}) = \{\mathbf{x} \in \mathbb{R}^d \ : \ (\mathbf{x} - \mathbf{m})^T A(\mathbf{x} - \mathbf{m}) = 1\}, \quad \text{where } A = K_{\mathbf{XX}}^{-1}$$

2 Ellipsoidal approximation

- By properties of covariance $K_{\mathbf{XX}}$ is (symmetric) positive semi-definite
- We assume $K_{\mathbf{XX}}$ is (symmetric) positive definite
- Then $\mathbf{m}$ and $K_{\mathbf{XX}}$ can be used to define an ellipsoid as the set of points:

$$\mathcal{E}_A(\mathbf{m}) = \{\mathbf{x} \in \mathbb{R}^d \ : \ (\mathbf{x} - \mathbf{m})^T A(\mathbf{x} - \mathbf{m}) = 1\}, \quad \text{where } A = K_{\mathbf{XX}}^{-1}$$

- Inverse used as the eigenvalues of A are the reciprocals of the squares of the semi-axis lengths

2 Finding the characteristic matrix

- We would like to find as robust approximation for $K_{\mathbf{XX}}$ as possible, which the direct computation might not be

2 Finding the characteristic matrix

- We would like to find as robust approximation for $K_{\mathbf{XX}}$ as possible, which the direct computation might not be
- Consider two sets of d-dimensional points $\mathbf{x} \in X$ and $\mathbf{y} \in Y$

2 Finding the characteristic matrix

- We would like to find as robust approximation for $K_{\mathbf{XX}}$ as possible, which the direct computation might not be
- Consider two sets of d-dimensional points $\mathbf{x} \in X$ and $\mathbf{y} \in Y$
- We can define a robust approximation of $K_{\mathbf{XX}}$ as the one whose inverse defines an ellipsoidal surface, which maximizes the distance to the closest points in both sets X and Y, whilst encompassing as many points in X as possible.

2 Finding the characteristic matrix

- We can use an algorithm inspired by *Support Vector Machines* to find the robust approximation

2 Finding the characteristic matrix

- We can use an algorithm inspired by *Support Vector Machines* to find the robust approximation
- If ellipsoid $\mathcal{E}_A(\mathbf{m})$ is chosen to encompass points $\mathbf{x} \in X$, but not $\mathbf{y} \in Y$ holds:

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

2 Finding the characteristic matrix

- We can use an algorithm inspired by *Support Vector Machines* to find the robust approximation
- If ellipsoid $\mathcal{E}_A(\mathbf{m})$ is chosen to encompass points $\mathbf{x} \in X$, but not $\mathbf{y} \in Y$ holds:

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

- We can add a buffer zone for robustness

$$(\mathbf{x}_i - \mathbf{m})^T A(\mathbf{x}_i - \mathbf{m}) - 1 \leq -1 \quad \text{and} \quad (\mathbf{y}_i - \mathbf{m})^T A(\mathbf{y}_i - \mathbf{m}) - 1 > 1$$

2 Finding the characteristic matrix

- We can use an algorithm inspired by *Support Vector Machines* to find the robust approximation
- If ellipsoid $\mathcal{E}_A(\mathbf{m})$ is chosen to encompass points $\mathbf{x} \in X$, but not $\mathbf{y} \in Y$ holds:

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

- We can add a buffer zone for robustness

$$(\mathbf{x}_i - \mathbf{m})^T A(\mathbf{x}_i - \mathbf{m}) - 1 \leq -1 \quad \text{and} \quad (\mathbf{y}_i - \mathbf{m})^T A(\mathbf{y}_i - \mathbf{m}) - 1 > 1$$

- And then relax this expression with non-negative variables $u_1, \ldots, u_{|X|}$ and $v_1, \ldots, v_{|Y|}$

$$(\mathbf{x}_i - \mathbf{m})^T A(\mathbf{x}_i - \mathbf{m}) \leq u_i, \quad \forall i \in \{1, \ldots, |X|\}$$
$$(\mathbf{y}_i - \mathbf{m})^T A(\mathbf{y}_i - \mathbf{m}) - 2 > -v_i, \quad \forall i \in \{1, \ldots, |Y|\}$$

2 Finding the characteristic matrix

- Then we would like to minimize the variables $u_1, \ldots, u_{|X|}$ and $v_1, \ldots, v_{|Y|}$, giving us an optimization problem formulation:

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

2 Extending to multiple ellipsoids

- Thus far we haven't given much thought for classification

2 Extending to multiple ellipsoids

- Thus far we haven't given much thought for classification
- Consider a single set of points $\mathbf{f} \in F$ belonging to different labels $l \in L_0$ and a notational mapping $c(\cdot)$ s.t. $c(\mathbf{f}) \in L_0$

2 Extending to multiple ellipsoids

- Thus far we haven't given much thought for classification
- Consider a single set of points $\mathbf{f} \in F$ belonging to different labels $l \in L_0$ and a notational mapping $c(\cdot)$ s.t. $c(\mathbf{f}) \in L_0$
- We can divide F into two sets consisting of points $\mathbf{x} \in X \ : \ c(\mathbf{x}) = l_x$ and $\mathbf{y} \in Y \ : \ c(\mathbf{y}) \in L_0 \setminus \{l_x\}$

2 Extending to multiple ellipsoids

- Thus far we haven't given much thought for classification
- Consider a single set of points $\mathbf{f} \in F$ belonging to different labels $l \in L_0$ and a notational mapping $c(\cdot)$ s.t. $c(\mathbf{f}) \in L_0$
- We can divide F into two sets consisting of points $\mathbf{x} \in X \ : \ c(\mathbf{x}) = l_x$ and $\mathbf{y} \in Y \ : \ c(\mathbf{y}) \in L_0 \setminus \{l_x\}$
- Choice of l_x is arbitrary

2 Extending to multiple ellipsoids

- Thus far we haven't given much thought for classification
- Consider a single set of points $\mathbf{f} \in F$ belonging to different labels $l \in L_0$ and a notational mapping $c(\cdot)$ s.t. $c(\mathbf{f}) \in L_0$
- We can divide F into two sets consisting of points $\mathbf{x} \in X \ : \ c(\mathbf{x}) = l_x$ and $\mathbf{y} \in Y \ : \ c(\mathbf{y}) \in L_0 \setminus \{l_x\}$
- Choice of l_x is arbitrary
- Hence we can find an ellipsoid for each label $l \in L_0$

2 Kernel regression as classification

Reminder of *semantic output classifier*

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

2 Kernel regression as classification

Reminder of *semantic output classifier*

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

- Define an additional random vector $\mathbf{S} = [S_1 \quad \ldots \quad S_p]^T$ for modeling the semantic vectors and random variable C for labels

2 Kernel regression as classification

Reminder of *semantic output classifier*

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

- Define an additional random vector $\mathbf{S} = [S_1 \quad \ldots \quad S_p]^T$ for modeling the semantic vectors and random variable C for labels
- Mapping $\mathcal{S}$ can be considered as a conditional expectation $\mathbb{E}[\mathbf{S}|C = c(\mathbf{f})]$, which written with probabilities gives:

2 Kernel regression as classification

Reminder of *semantic output classifier*

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

- Define an additional random vector $\mathbf{S} = [S_1 \quad \dots \quad S_p]^T$ for modeling the semantic vectors and random variable C for labels
- Mapping $\mathcal{S}$ can be considered as a conditional expectation $\mathbb{E}[\mathbf{S}|C = c(\mathbf{f})]$, which written with probabilities gives:

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

2 Kernel regression as classification

- Using kernel density estimation we can rewrite this as:

$$\mathcal{S}(\mathbf{f}) = \sum_{\mathbf{s} \in S} \frac{\mathbb{P}[\mathbf{S} = \mathbf{s} \cap C = c(\mathbf{f})]}{\mathbb{P}[C = c(\mathbf{f})]} \mathbf{s} = \frac{\sum_{\mathbf{s} \in S} K(d_{\mathbf{s}}(\mathbf{f}))\mathbf{s}}{\sum_{\mathbf{s} \in S} K(d_{\mathbf{s}}(\mathbf{f}))} = \mathbf{s}_0$$

2 Kernel regression as classification

- Using kernel density estimation we can rewrite this as:

$$\mathcal{S}(\mathbf{f}) = \sum_{\mathbf{s}\in S} \frac{\mathbb{P}[\mathbf{S} = \mathbf{s} \cap C = c(\mathbf{f})]}{\mathbb{P}[C = c(\mathbf{f})]}\,\mathbf{s} = \frac{\sum_{\mathbf{s}\in S} K(d_{\mathbf{s}}(\mathbf{f}))\mathbf{s}}{\sum_{\mathbf{s}\in S} K(d_{\mathbf{s}}(\mathbf{f}))} = \mathbf{s}_0$$

- This is essentially Nadaraya-Watson kernel regression

2 Kernel regression as classification

- Using kernel density estimation we can rewrite this as:

$$\mathcal{S}(\mathbf{f}) = \sum_{\mathbf{s}\in S} \frac{\mathbb{P}[\mathbf{S}=\mathbf{s}\cap C=c(\mathbf{f})]}{\mathbb{P}[C=c(\mathbf{f})]}\mathbf{s} = \frac{\sum_{\mathbf{s}\in S} K(d_{\mathbf{s}}(\mathbf{f}))\mathbf{s}}{\sum_{\mathbf{s}\in S} K(d_{\mathbf{s}}(\mathbf{f}))} = \mathbf{s}_0$$

- This is essentially Nadaraya-Watson kernel regression
- $K(\cdot)$ is the kernel function and $d_{\mathbf{s}}(\cdot)$ defines the distance metric

2 Kernel regression as classification

- Using kernel density estimation we can rewrite this as:

$$\mathcal{S}(\mathbf{f}) = \sum_{\mathbf{s}\in S} \frac{\mathbb{P}[\mathbf{S} = \mathbf{s} \cap C = c(\mathbf{f})]}{\mathbb{P}[C = c(\mathbf{f})]}\,\mathbf{s} = \frac{\sum_{\mathbf{s}\in S} K(d_{\mathbf{s}}(\mathbf{f}))\mathbf{s}}{\sum_{\mathbf{s}\in S} K(d_{\mathbf{s}}(\mathbf{f}))} = \mathbf{s}_0$$

- This is essentially Nadaraya-Watson kernel regression
- $K(\cdot)$ is the kernel function and $d_{\mathbf{s}}(\cdot)$ defines the distance metric
- Natural choice for distance metric is the *Mahalanobis distance*

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

2 Kernel regression as classification

- Using kernel density estimation we can rewrite this as:

$$\mathcal{S}(\mathbf{f}) = \sum_{\mathbf{s}\in S} \frac{\mathbb{P}[\mathbf{S}=\mathbf{s}\cap C=c(\mathbf{f})]}{\mathbb{P}[C=c(\mathbf{f})]}\,\mathbf{s} = \frac{\sum_{\mathbf{s}\in S} K(d_{\mathbf{s}}(\mathbf{f}))\mathbf{s}}{\sum_{\mathbf{s}\in S} K(d_{\mathbf{s}}(\mathbf{f}))} = \mathbf{s}_0$$

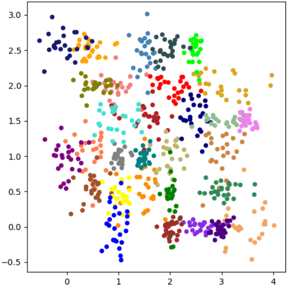
- This is essentially Nadaraya-Watson kernel regression
- $K(\cdot)$ is the kernel function and $d_{\mathbf{s}}(\cdot)$ defines the distance metric
- Natural choice for distance metric is the *Mahalanobis distance*

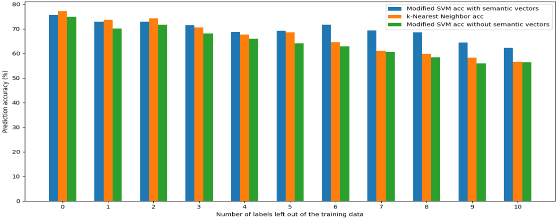
$$d_M(\mathbf{v},\mathbf{X}) = \sqrt{(\mathbf{v}-\mathbf{m})^T K_{\mathbf{XX}}^{-1}(\mathbf{v}-\mathbf{m})}$$

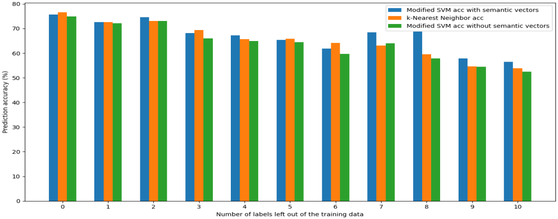
- As S^P consists of *one-hot encodings* of labels the wanted label is found as the one which semantic vector is closest to our approximation

3
Results

3 Randomized points







3 MNIST Dataset

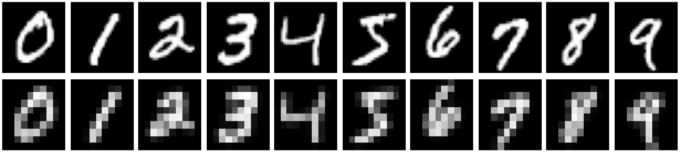


Table: Prediction accuracies for our algorithm and KNN for 5 different samples with 400 training points and 200 testing points per label

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