VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Algorithms used for classifying visual signals using methods for extracting significant features

Algoritmy pro klasifikaci visuálních signálů s využitím technik extrakce významných rysů

Bc. Vojtěch Dorňák

Diploma Thesis

Supervisor: Ing. Lukáš Pospíšil, Ph.D.

Ostrava, 2021

Abstrakt

Tohle je český abstrakt

Klíčová slova

typografie, LaTeX, diplomová práce

Abstract

This is English abstract.

Keywords

typography, LaTeX, master thesis

Acknowledgement

I would like to thank all those who helped me with the work, because without them this work would not have happened.

Contents

List of symbols and abbreviations **5**

1 Introduction **6**

2 Image Data Transformation **7**

2.1 Scale Invariant Feature Transform 7

2.2 Speeded Up Robust Features 11

2.3 Oriented FAST and Rotated BRIEF 13

2.4 Bag of Visual Words 15

2.5 Principal Component Analysis 17

3 Classifiers **20**

3.1 Bayesian Model 20

3.2 The Support Vector Machine 22

3.3 Metrics 26

4 Datasets **27**

5 Results **28**

6 Conclusion **29**

Bibliography **30**

List of symbols and abbreviations

SIFT	–	Scale Invariant Feature Transform
SURF	–	Speed Up Robust Features
ORB	–	Oriented Fast and Rotated Brief
SVM	–	Support Vector Machine

Chapter 1

Introduction

The goal of this thesis...

Chapter 2

Image Data Transformation

We can attempt to classify the raw image data where the image is represented as a vector of light intensities at each pixel. However, it might be beneficial to transform the image data into a feature space using a feature extraction technique. The feature extraction techniques can be split into two categories: local feature extractors and global feature extractors[1].

Local feature extractors localize such points in an image, which could be found regardless of the image subject position. These points are called key-points. The area around such key-points is then described using a vector called a descriptor. The descriptors are created in a way, which allows for matching similar features.

Global feature extractors transform the whole image into a lower-dimensional vector, attempting to retain the most important information.

For feature extraction, we compare three local feature extractors: SIFT, SURF and ORB, and a global feature extractor: PCA.

In this section, we describe the four feature extraction algorithms.

2.1 Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) was proposed by David Lowe, and published in the original paper [2] in 1999.

Compared to ordinary techniques such as the Harris Corner Detector [3] or Canny edge detector [4], the SIFT identifies the general (not only edges and corners) key-points. The descriptors, which describe the local image region around the key-points are scale-invariant. Moreover, they are invariant to rotation, illumination, and change in affine transformations. In our text, we introduce the methodology of training the classifier that recognizes the objects in real images (photographs); therefore the properties of SIFT feature vector are essential.

2.1.1 Key-point Detection

Let us consider an input image $I_{img}(x, y)$. A convolution of the image with a Gaussian kernel

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.1}$$

at a scale σ is exploited to get a scale-space representation of the image so that:

$$L(x, y, \sigma) = G(x, y, \sigma) * I_{img}(x, y).$$

To detect scale-invariant key-points, a scale normalized Laplacian of Gaussian (LoG)

$$\triangle_{norm} L(x, y, \sigma) = \sigma \left(\frac{\partial^2 L(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 L(x, y, \sigma)}{\partial y^2} \right)$$

is required[5]. It has been shown[6], that the local extrema of LoG provide the most stable image features, compared to many other popular image functions.

Determination of the LoG would be time-consuming, therefore, it is approximated by the Difference of Gaussian (DoG)[7]. The DoG is computed from two adjacent scales separated by a constant multiplicative factor $k \in \mathbb{R}$:

$$D(x, y, \sigma) := L(x, y, k\sigma) - L(x, y, \sigma).$$

To ensure scale invariance, the extrema are located not only in the domain of one DoG, however, it is found across different scales as well. The different DoGs at the scales are created by progressively convolving the original image with the Gaussian kernel. The consecutive Gaussian kernels' scale differs by the multiplicative factor k.

At each doubling of the scale k, the resolution of an image can be reduced by a factor of 2 for efficiency. Each group of blurred images of the same resolution is called an octave. In each octave, we generate the DoG by subtracting the $L(x, y, \sigma)$ of neighboring scales. This is called the DoG pyramid and can be seen in Figure 2.1.

Each pixel in the DoG pyramid is compared to the 3×3 pixels in DoGs below and above, and the 8 surrounding pixels. This is shown in Figure 2.2. The pixel is selected as a key-point candidate if it has a lower or a higher value than all of these 26 pixels.

To detect sub-pixel locations of extrema, the DoG is interpolated using quadratic Taylor expansion at each key-point candidate:

$$D(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}} \boldsymbol{x} + \frac{1}{2} \boldsymbol{x}^T \frac{\partial^2 D}{\partial \boldsymbol{x}^2} \boldsymbol{x},$$

where $D(x, y, \sigma)$ and the partial derivatives are evaluated at the key-point and $\boldsymbol{x} = (x, y, \sigma)$ is the

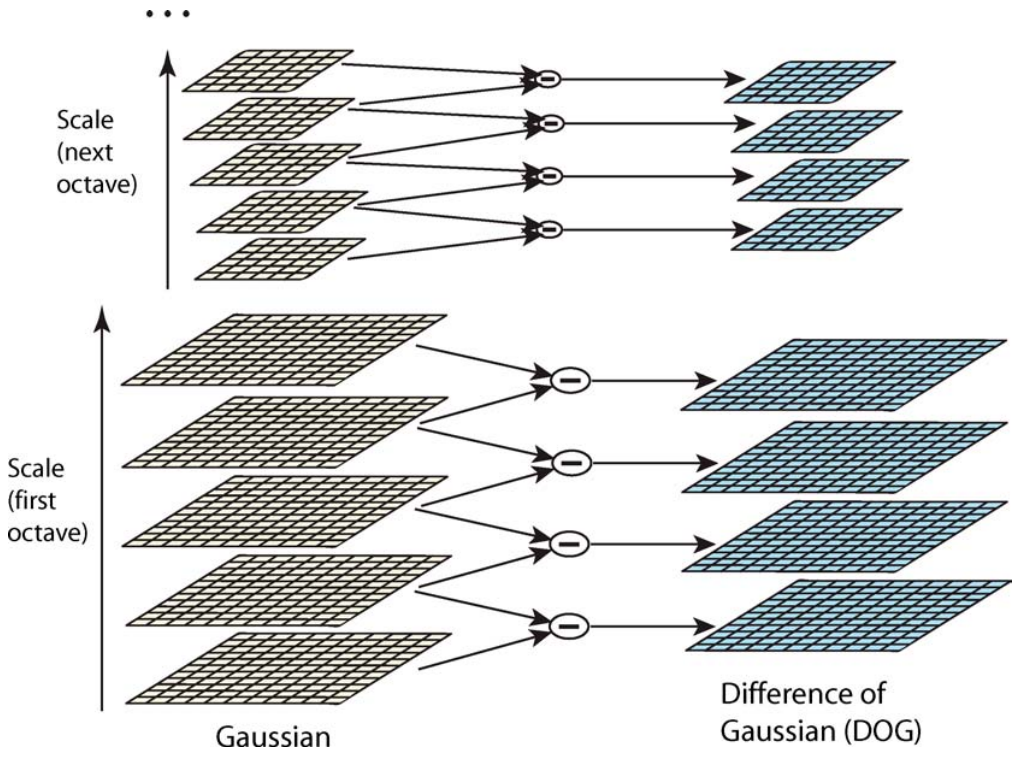


Figure 2.1: DoG pyramid. [7]

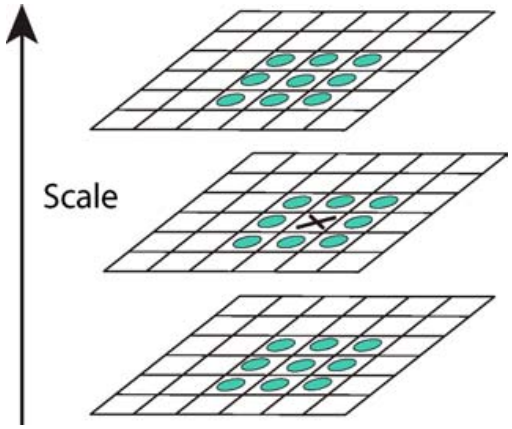


Figure 2.2: Optima of the DoG are selected by the means of comparing pixel to the 26 pixels surrounding it in the scale-space. [7]

offset from the key-point. Then, extremum $\hat{\boldsymbol{x}}$ is located by setting the gradient of D to be a zero vector:

$$\hat{\boldsymbol{x}} = -\frac{\partial^2 D^{-1}}{\partial \boldsymbol{x}^2}\frac{\partial D}{\partial \boldsymbol{x}}.$$

If the offset $\hat{\boldsymbol{x}}$ is larger than 0.5 in any dimension, it indicates that the extremum is closer to another point. In this case, the candidate point is changed to the new point, and interpolation is performed again.

The function value at the sub-pixel extrema ,$D(\hat{\boldsymbol{x}})$, is used for rejecting extrema with low contrast. In our experiments, we use the OpenCV SIFT implementation default value $\frac{0.04}{3}$[8].

The DoG has a strong response along edges, even if the location along the edge is poorly determined. These candidate key-points have principal curvature perpendicular to the edge much larger than the principal curvature along it. The principal curvatures can be computed from a 2×2 Hessian matrix at the location of the key-point candidate

$$\boldsymbol{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix},$$

where the partial derivatives are estimated by taking the differences of neighboring sample points. The eigenvalues of $\boldsymbol{H}$ are proportional to the principal curvatures.

The computation of eigenvalues can be avoided, as only the ratio of the eigenvalues is required. Let us denote the larger eigenvalue as α and the smaller one as β. The trace of $\boldsymbol{H}$ is defined as

$$Tr(\boldsymbol{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

and the determinant as

$$Det(\boldsymbol{H}) = D_{xx}D_{yy} - D_{xy}^2 = \alpha\beta.$$

Let r be the ratio of the eigenvalues, such that

$$r = \frac{\alpha}{\beta}.$$

Then,

$$\frac{Tr(H)^2}{Det(\boldsymbol{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}.$$

Therefore, to inspect the ratio of eigenvalues, we can check

$$\frac{Tr(H)^2}{Det(\boldsymbol{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}.$$

The candidate key-points with this ratio below a threshold r is discarded. In our experiments, we use the default value of OpenCV SIFT implementation: 10.

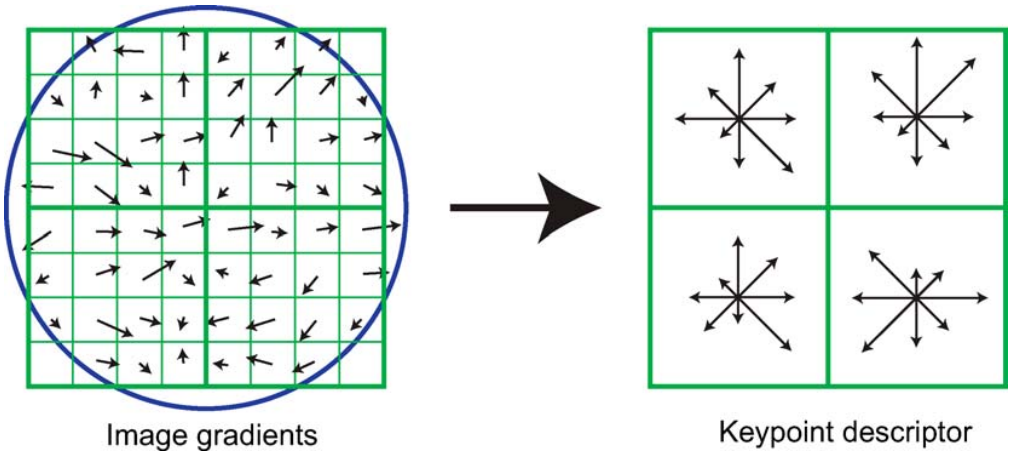


Figure 2.3: Building of the sift-descriptor. [7]

2.1.2 Key-point Description

We want to create a descriptor for each of our key-points. These descriptors must be almost identical across different scales, rotations, illuminations, and other transformations.

To ensure descriptor invariance to a rotation, we first determine the key-point orientation. First, we calculate the gradient magnitude

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2},$$

and orientation

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right),$$

of each point within a region around the key-point. From these, an orientation histogram with 36 bins is created. The largest peak is selected as a key-point orientation. If there are other peaks more than 80% of the largest peak, new key-points are created at the same location with the other peaks as their orientation.

For the descriptor, a window, 16×16 pixels around the key-point, rotated by the orientation of the key-point, is used. In this window, the gradient magnitude and the orientation are computed for each point. The 16×16 window is divided into 16 (4×4) sub-windows. For each sub-window, an 8 bin gradient orientation histogram, weighted by gradient magnitudes, is created. This can be seen for a smaller 8×8 window in Figure 2.3. These histograms then form a descriptor vector.

2.2 Speeded Up Robust Features

Speeded Up Robust Features (SURF) is a local feature transform algorithm proposed by Herbert Bay, Tinne Tuytelaars, and Luc Van Gool in 2006[9]. Compared to SIFT[2], the authors claim the

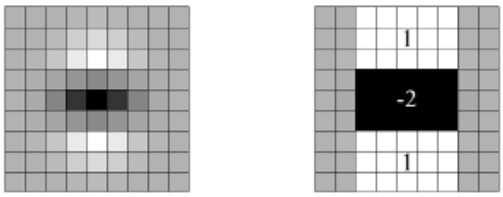


Figure 2.4: Gaussian second order derivative in the y-direction and its approximation using box filters [9]

SURF detector and descriptor, to be faster and more robust against various image transformations.

2.2.1 Key-point Detection

The SURF key-point detection is based on the determinant of the Hessian matrix. Let us consider an input image $I_{img}(x, y)$ and the scale-space representation of the image

$$L(x, y, \sigma) = G(x, y, \sigma) * I_{img}(x, y),$$

where $G(x, y, \sigma)$ is the Gaussian kernel described in (2.1).

The Hessian matrix at scale σ is defined as

$$\mathcal{H}(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix},$$

where $L_{xx}(x, y, \sigma)$, $L_{xy}(x, y, \sigma)$ and $L_{yy}(x, y, \sigma)$ are the second-order derivatives of the scale-space representation of the image at a point (x, y).

As the property of Gaussian filter, that no new structures can appear while going to a lower resolution has been shown not to apply in 2D case [5], the SURF authors choose to approximate the second-order derivatives of Gaussian filter with box filters (shown in Figure 2.4). These allow for the use of integral images, which reduce the computational complexity.

Let us denote the approximations by D_{xx}, D_{xy}, and D_{yy}. The relative weights in the calculation of determinant of Hessian need to be weighted by 0.9, which yields

$$\det(\mathcal{H}) = D_{xx} * D_{yy} - (0.9 * D_{xy})^2.$$

Due to the use of box filters and integral images, any size of the box filter can be applied to the original image at the same speed directly. Therefore, the scale space is created by the use of up-scaled filters of sizes 9×9, 15×15, 21×21, 27×27, etc. For each octave, the difference between filter sizes is doubled (from 6 to 12 to 24).

As the box filter layout remains the same, the corresponding Gaussian filter scales accordingly. The 9×9 box filter corresponds to a Gaussian filter with the scale $\sigma = 1.2$. From this, we can calculate the corresponding Gaussian filter scale for each box filter size.

To select the key-points, non-maximum suppression in the $3 \times 3 \times 3$ neighborhood of each point is applied. The maxima of the determinant of the Hessian matrix are then interpolated the same way as in the SIFT algorithm (using quadratic Taylor expansion).

2.2.2 Key-point Description

First, we need to ensure the descriptor's invariance to rotation. For the key-point, we calculate the Haar-wavelet responses in the x and the y direction in a circular neighborhood of $6s$, where s is the Gaussian filter scale at which the key-point was found. Integral images are used for speeding up the process.

The wavelet responses are then weighted with a Gaussian($\sigma = 2.5s$) centered at the key-point. The weighted responses are represented as vectors in a space with the x response being a vector along the x-axis, and the y response being a vector along the y-axis. All the vectors within a sliding window of $\frac{\pi}{3}$ are summed, and the longest of the vector is selected as the key-point orientation.

The descriptor is constructed from a square window the size of $20s$ around a key-point. The window is then rotated along with the key-point orientation. This window is then divided into 16 (4×4) square sub-windows. In each sub-window, the features are calculated using 5×5 regularly spaced sample points. For these, we calculate the Haar wavelet responses in the horizontal and the vertical direction, where "horizontal" and "vertical" are defined in relation to the key-point orientation. The responses are weighted with a Gaussian($\sigma = 3.3s$), centered at the key-point.

Let us call the horizontal responses d_x and the vertical responses d_y. Over each sub-window, we denote the sum of the responses $\sum d_x$ and $\sum d_y$, and the sum of absolute values of the responses $\sum |d_x|$ and $\sum |d_y|$. The behavior of these values for different image patterns can be seen in Figure 2.5. Combining $\sum d_x$, $\sum d_y$, $\sum |d_x|$, and $\sum |d_y|$ for each of the 16 sub-window into a vector, we get a descriptor of length 64.

2.3 Oriented FAST and Rotated BRIEF

Oriented FAST and Rotated BRIEF (ORB) is a local feature extractor proposed by Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski in 2011 [10].

The algorithm builds on the FAST (Features from Accelerated Segment Test) corner detector[11] and the BRIEF (Binary Robust Independent Elementary Features) descriptor[12].

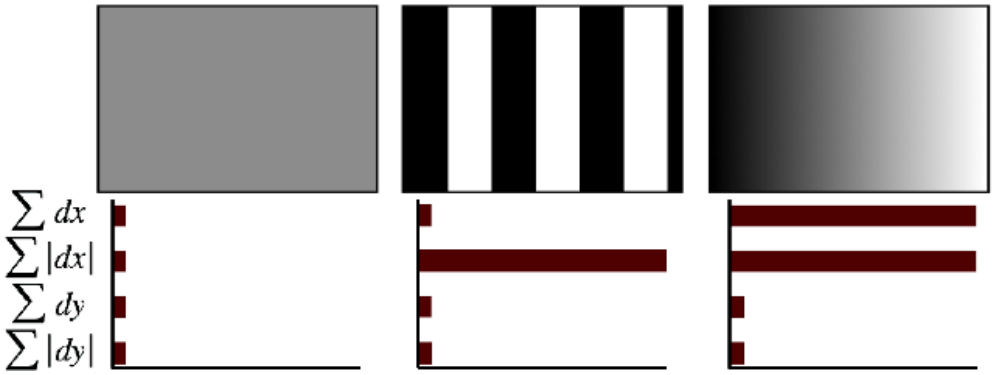


Figure 2.5: Behaviour of SURF descriptor for different image patterns[9]

2.3.1 Key-point Detection

The ORB algorithm uses the FAST-9 variant of the FAST corner detector. This detector compares each pixel intensity (denoted as I_p) in an image with the intensities of pixels in a circle of radius 9 around the pixel.

Let $n \in \mathbb{N}$ and the threshold $t \in \mathbb{R}^+$ be given. Let S be a set of pixels in a circle of radius 9 around the examined pixel. Let us denote I_x as the intensity of a pixel x. The examined pixel is selected as a corner if there exists a set $S_n \in S$ of n contiguous pixels, where $\forall x \in S_n : I_x + t < I_p$, or $\forall x \in S_n : I_x - t > I_p$.

The selected corners, i.e. key-points are then ordered according to a Harris corner measure[3]. For N key-points, the threshold is selected low enough to get more than N key-points. The best N key-points (according to the Harris corner measure) are then selected.

To find multi-scale features, the scale pyramid of an image is generated and key-points are generated at each level in the pyramid.

2.3.2 Key-point Description

To ensure the descriptor's invariance to rotation, we need to determine the key-point orientation. For this, the intensity centroid[13] is used. The intensity centroid expects the intensity of a key-point to be offset from its center. The vector from the center to the centroid is used for the key-point orientation.

Given image $I(x, y)$, the centroid is found using a moment of a patch

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y).$$

The centroid is then located as

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right),$$

from which we can obtain the key-point orientation

$$\theta = \text{atan2}(m_{01}, m_{10}),$$

where atan2 is the quadrant-aware version of arctan.

As an ORB descriptor, a variation on the BRIEF descriptor is used. The BRIEF descriptor is a vector of binary values. This allows for fast matching using a Hamming distance.

The descriptor values are computed by comparing random pairs of pixel intensities in a patch. The binary vector is created from the responses on a patch $\boldsymbol{p}$ of test

$$\tau(\boldsymbol{p}; x, y) = \begin{cases} 1 & \text{if } \boldsymbol{p}(x) < \boldsymbol{p}(y), \\ 0 & \text{otherwise,} \end{cases}$$

where $\boldsymbol{p}(x)$ is the pixel intensity of a patch $\boldsymbol{p}$ at a point x. The pairs of points x and y are from a random predetermined set

$$\boldsymbol{S} = \begin{pmatrix} x_1 \dots x_n \\ y_1 \dots y_n \end{pmatrix},$$

where n is the size of our descriptor. The BRIEF descriptor is then defined as a vector of n binary tests

$$f_n(\boldsymbol{p}) \coloneqq \sum_{i=1}^{n} 2^{i-1} \tau(\boldsymbol{p}; x_i, y_i).$$

The steered version of the BRIEF descriptor, according to the orientation θ of the key-point, can be created by using a corresponding rotation matrix $\boldsymbol{R}_\theta$:

$$\boldsymbol{S}_{\boldsymbol{\theta}} = \boldsymbol{R}_\theta \boldsymbol{S}.$$

The sets of pairs are precomputed in a lookup table for discretized θ in increments of $\frac{2\pi}{30}$ to improve speed. In the end, the steered BRIEF descriptor becomes

$$g_n(\boldsymbol{p}, \theta) \coloneqq f_n(\boldsymbol{p}) | (x_i, y_i) \in \boldsymbol{S}_\theta.$$

2.4 Bag of Visual Words

Local feature extractors provide us with varied number of descriptors for each image. However, for classification, we need each image to be represented by a single vector. This can be achieved using a Bag of Words technique.

Bag of Words (BoW) is a technique, which represents each sample in a dataset as a multiset of its words. In general, these words do not have to be literal words but can be any categorization of information provided by the sample.

First BoW needs to generate a dictionary. The dictionary is a set of $k \in \mathbb{N}$ categories of words called bins. The BoW then assigns each word to a bin. Finally, for each sample in the dataset, a histogram of amounts of words belonging to each bin is created.

We explain the technique with text documents as samples. With text documents, the dictionary consists of every unique word in all documents. The words of each document are assigned to a bin of said words.

For example, in a dictionary consisting of bins { "Peter", "run", "talked", "to" }, a sample sentence "Peter talked to Peter" would produce the following histogram:

$$\{2, 0, 1, 1\}.$$

In our case, we consider the descriptors found in an image as words. As we are dealing with visual information, let us call these words "visual words" and nickname the BoW technique "Bag of Visual Words" (BoVW).

We generate the categories using a k-means algorithm on all the visual words in the training dataset. We then assign the visual words to the closest (considering euclidean distance) category.

2.4.1 k-means

The k-means algorithm is an unsupervised learning technique. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a dataset of n data-points. The goal is to sort the data points into k clusters $S = \{S_1, S_2, \ldots, S_k\}$, such that

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2 ,$$

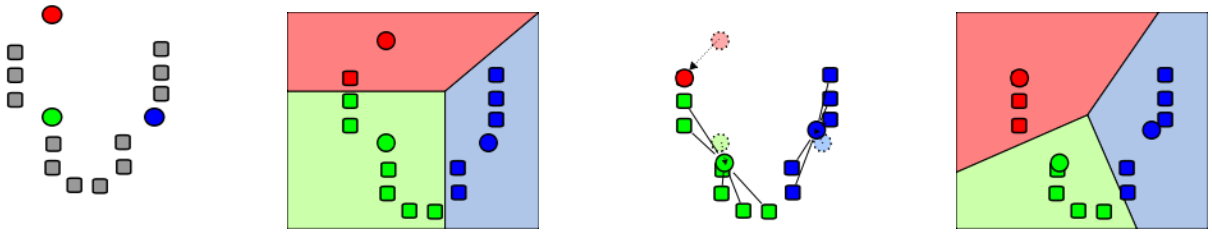
where $c_1, c_2, ..., c_k$ are the centroids of $S_1, S_2, ..., S_k$, respectively. The centroids are determined as means of data points belonging to each cluster.

The algorithm starts by selecting k random data points as the centroids. Then alternates between these two steps until a terminate condition is met:

Assignment step: Each data point is assigned to the cluster, such that the Euclidean distance between the centroids of the clusters and the current data point is the smallest one.

Update step: For each cluster, a new centroid is computed as the mean of data points belonging to this cluster.

The terminate condition is met, when the ratio of the samples, for which the assigned cluster changes, to the total amount of samples, is smaller than a specified threshold. The steps of the algorithm can be seen in Figure 2.6. The challenge is determining the k number of clusters, and therefore the size of the visual dictionary. In our experiments, we try different settings for k to find a clustering, which best represents our data.



(a) Initial centroids are selected.

(b) By choosing the nearest centroid, each data point is assigned its cluster.

(c) New centroids are calculated as means of data points in the cluster.

(d) Steps Figure 2.6b and Figure 2.6c are repeated.

Figure 2.6: Lloyd algorithm demonstration. [14]

2.5 Principal Component Analysis

Principal Component Analysis (PCA) is a global feature extractor. It is used to reduce the dimension of data while preserving as much of the data variance as possible.

Let $x_t \in \mathbb{R}^n$ be a data point, where $t = 1, \ldots, T$ and T is the number of data points. We want to project x_t into a $y_t \in \mathbb{R}^k$, where $k < n$. We want to find the optimal parameters P of the parametric reduction mapping $\psi_P : \mathbb{R}^n \to \mathbb{R}^k$ and the parametric reconstruction mapping $\psi_P^{-1} : \mathbb{R}^k \to \mathbb{R}^n$. The optimal parameters P^* minimize the reconstruction error, i.e.,

$$P^* := \arg\min_{P \in \rho} \sum_{t=1}^{T} \left\| x_t - \psi_P^{-1}(\psi_P(x_t)) \right\|, \tag{2.2}$$

where ψ denotes the set of feasible parameters and $\left\| x_t - \psi_P^{-1}(\psi_P(x_t)) \right\|$ is the reconstruction error of a data-point.

PCA uses the mappings

$$\psi_{[Q,b]}^{-1}(y) := Qy + b, \psi_{[Q,b]}(x) := Q^\top (x - b), \tag{2.3}$$

with parameters $b \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n,k}$ with orthonormal columns, i.e.,

$$Q^\top Q = I \in \mathbb{R}^{k,k}. \tag{2.4}$$

Substituting (2.3), (2.4) into the optimization problem (2.2) and using the square Euclidean norm for measuring the reconstruction error, we get the optimization problem

$$[Q^*, b^*] := \arg\min_{Q,b} \sum_{t=1}^{T} \left\| x_t - (QQ^\top (x_t - b) + b) \right\|_2^2 \text{s.t.} Q^\top Q = I. \tag{2.5}$$

The objective function $f(Q, b)$ of (2.5) can be rewritten to the form

$$f(Q,b) = \sum_{t=1}^{T}(x_t^\top x_t - 2x_t^\top b + b^\top b - x_t^\top QQ^\top x_t + 2x_t^\top QQ^\top b - b^\top QQ^\top b), \tag{2.6}$$

and the optimality condition of b^* can be formulated as

$$\nabla_b f(Q,b) = \sum_{t=1}^{T}(-2x_t + 2b + 2QQ^\top xt - 2QQ^\top b) = 0,$$

which is equivalent to

$$(I - QQ^\top)\sum_{t=1}^{T}(b - x_t) = 0. \tag{2.7}$$

As $k < n$, $Q \in \mathbb{R}^{n,k}$ is not a full column rank and $QQ^\top \neq= I$. The unique solution of (2.7) is

$$b^* = \frac{1}{T}\sum_{t=1}^{T} x_t.$$

Moreover, the Hessian matrix

$$\nabla^2_{b,b} f(Q,b) = 2(I - QQ^\top), \tag{2.8}$$

is a symmetric positive definitive matrix, therefore the objective function (2.6) is strictly convex (in the variable b) and (2.8) is a unique minimizer.

Let us denote the shifted data by

$$\hat{x_t} \coloneqq x_t - b^*, t = 1, \ldots, T$$

and write the objective function of (2.6) as

$$f(Q,b^*) = \sum_{t=1}^{T}\left\|\hat{x_t} - QQ^\top \hat{x_t}\right\|_2^2 = \sum_{t=1}^{T}(\hat{x_t}^\top \hat{x_t} - \hat{x_t}^\top QQ^\top \hat{x_t}). \tag{2.9}$$

We can simplify (2.9) using properties of the matrix trace:

$$f(Q,b^*) = \sum_{t=1}^{T}\mathrm{trace}(\hat{x_t}^\top \hat{x_t}) - \sum_{t=1}^{T}\mathrm{trace}(\hat{x_t}^\top QQ^\top \hat{x_t}) = \mathrm{trace}(\mathrm{cov}(x)) - \mathrm{trace}(Q^\top \mathrm{cov}(x)Q),$$

where $\mathrm{cov}(x)$ is the covariance matrix of x defined as

$$\mathrm{cov}(x) \coloneqq \sum_{t=1}^{T}(x_t - b^*)(x_t - b^*)^\top = \sum_{t=1}^{T}\hat{x_t}\hat{x_t}^\top.$$

The argument of the minimum is independent of constants in the objective function. Therefore, we can reformulate the optimization problem (2.5) (in terms of variable Q) as

$$\begin{aligned}[Q^*] &= \underset{Q^\top Q=I}{\arg\min}\, f(Q, b^*) = \underset{Q^\top Q=I}{\arg\min} -\text{trace}(Q^\top \text{cov}(x) Q) \\ &= \underset{Q^\top Q=I}{\arg\max}\, \text{trace}(Q^\top \text{cov}(x) Q) \\ &= \underset{\forall j: q_j^\top q_j = 1}{\arg\max} \sum_{j=1}^{k} q_j^\top \text{cov}(x) q_j,\end{aligned} \tag{2.10}$$

where $q_j, j = 1, \ldots, k$ denote the (orthonormal) columns of the matrix $Q \in \mathbb{R}^{n,k}$. The Lagrange function corresponding to the problem (2.10) is given by

$$L(Q, \lambda) \coloneqq \sum_{j=1}^{k} q_j^\top \text{cov}(x) q_j - \sum_{j=1}^{k} \lambda_j (q_j^\top q_j - 1),$$

where $\lambda \in \mathbb{R}^K$ denotes Lagrange multiplier corresponding to equality constraints. The first Karush-Kuhn-Tucker conditions can be derived as

$$\nabla_{q_j} L(Q, \lambda) = 2\text{cov}(x) q_j - 2\lambda_j q_j = 0, j = 1, \ldots, k,$$

which are the eigenvalue equations

$$\text{cov}(x) q_j = \lambda_j q_j, j = 1, \ldots, k. \tag{2.11}$$

If we substitute (2.11) into objective function (2.10), we get

$$\sum_{j=1}^{k} q_j^\top \text{cov}(x) q_j = \sum_{j=1}^{k} \lambda_j q_j^\top q_j = \sum_{j=1}^{k} \lambda_j.$$

As the problem (2.10) is a maximization problem, the optimal Q^* consists of (orthonormal) eigenvectors which correspond to the k largest eigenvalues.

Chapter 3

Classifiers

For data classification we are using two classification techniques, the Bayes model and SVM. The goal of these techniques is to find a mapping from a feature space into a space of labels.

3.1 Bayesian Model

This classifier is suitable for classifying data represented by a stochastic vector. The BoVW data can be easily transformed into a such vector. Instead of each component of the BoVW vector representing the number of key-points in the respective category, the component in our new vector represents the probability of key-points belonging to the respective category.

Let us denote the stochastic data vector $\Pi_{xt} \in \mathbb{R}^{K_x}, t = 1, \ldots, T$, where T is the number of samples. Let the vector $\Pi yt \in \mathbb{R}^{K_y}$ be a vector of probabilities, with which Πxt belongs to each category, and K_y the number of categories.

Given a stochastic data vector Π_x, we can describe the transformation $\mathbb{R}^{K_x} \to \mathbb{R}^{K_y}$ using a matrix $\Delta \in \mathbb{R}^{K_y, K_x}$:

$$\Delta = \begin{bmatrix} P(\Pi_y^1|\Pi_x^1) & P(\Pi_y^1|\Pi_x^2) & \ldots & P(\Pi_y^1|\Pi_x^{K_x}) \\ P(\Pi_y^2|\Pi_x^1) & P(\Pi_y^2|\Pi_x^2) & \ldots & P(\Pi_y^2|\Pi_x^{K_x}) \\ \vdots & \ddots & & \\ P(\Pi_y^{K_y}|\Pi_x^1) & P(\Pi_y^{K_y}|\Pi_x^2) & \ldots & P(\Pi_y^{K_y}|\Pi_x^{K_x}) \end{bmatrix},$$

where Π_x^n is the n-th element of Π_x, similar to Π_y^n, and the matrix Δ is a left stochastic matrix.

The search for the optimal Δ^* can be written as

$$\Delta^* = \underset{\Delta \in \Omega_\Delta}{\arg\min} \sum_{t=1}^{T} \mathrm{dist}(\Pi_{yt}, \Delta \Pi_{xt}),$$

where Ω_Δ is a set of left stochastic matrices. The $\text{dist}(\Pi_{yt}, \Delta\Pi_{xt})$ is calculated as Kullback-Leiber divergence[15]:

$$\text{dist}(\Pi_{yt}, \Delta\Pi_{xt}) = -\sum_{i=1}^{K_y} \Pi_{yt}^i \ln \frac{(\Delta\Pi_{xt})_i}{\Pi_{yt}^i} = -\sum_{i=1}^{K_y} \Pi_{yt}^i (\ln(\Delta\Pi_{xt})_i - \ln \Pi_{yt}^i).$$

For the optimization, the term $\ln \Pi_{yt}^i$ is constant, therefore it can be ignored:

$$\text{dist}(\Pi_{yt}, \Delta\Pi_{xt}) \propto -\sum_{i=1}^{K_y} \Pi_{yt}^i \ln(\Delta\Pi_{xt})_i.$$

This problem is hard to minimize analytically. However, $\Delta\Pi_{xt}$ is a convex function, and therefore $-\ln(\Delta\Pi_{xt})$ is a convex function. Thus Jensen's inequality can be used:

$$-\sum_{i=1}^{K_y} \Pi_{yt}^i \ln(\Delta\Pi_{xt})_i \leq -\sum_{i=1}^{K_y} \Pi_{yt}^i (\sum_{j=1}^{K_x} \Pi_{xt}^j \ln(\Delta_{ij})) = -\sum_{i=1}^{K_y}\sum_{j=1}^{K_x} \Pi_{yt}^i \Pi_{xt}^j \ln \Delta_{ij}.$$

From this, we get an optimization problem

$$\Delta^* = \underset{\Delta \in \Omega_\Delta}{\arg\min} -\sum_{t=1}^{T}\sum_{i=1}^{K_y}\sum_{j=1}^{K_x} \Pi_{yt}^i \Pi_{xt}^j \ln \Delta_{ij},$$

where

$$\Omega_{Delta} = \{\Delta \in [0,1], \forall j \in \{1, 2, \ldots, K_x\} : \sum_{i=1}^{K_y} \Delta_{ij} = 1\},$$

which can be solved analytically.

Let Δ be the optimal stochastic matrix. There exist $\lambda_j \in \mathbb{R}^{K_x}$ such that

$$L(\Delta, \lambda) = -\sum_{t=1}^{T}\sum_{i=1}^{K_y}\sum_{j=1}^{K_x} \Pi_{yt}^i \Pi_{xt}^j \ln \Delta_{ij} + \sum_{j=1}^{K_x} \lambda_j (\sum_{i=1}^{K_y} \Delta_{ij} - 1)$$

is the Lagrange function. The Karush-Kuhn Tucker conditions are

$$\nabla_{\Delta_{\hat{i}\hat{j}}} L(\Delta_{\hat{i}\hat{j}}, \lambda) = -\frac{1}{\Delta_{\hat{i}\hat{j}}} \sum_{t=1}^{T} \Pi_{yt}^{\hat{i}} \Pi_{xt}^{\hat{j}} + \lambda_{\hat{j}} = 0, \tag{3.1}$$

and

$$\nabla_{\Delta_{\hat{j}}} = \sum_{i=1}^{K_y} \Delta_{i\hat{j}} - 1 = 0. \tag{3.2}$$

From (3.1) and (3.2), we get the optimal $\Delta^*_{\hat{i}\hat{j}}$

$$\Delta^*_{\hat{i}\hat{j}} = \frac{\sum_{t=1}^{T} \Pi^{\hat{i}}_{yt} \Pi^{\hat{j}}_{xt}}{\sum_{i=1}^{K_y} \sum_{t=1}^{T} \Pi^{i}_{yt} \Pi^{\hat{j}}_{xt}},$$

Another approach to finding the optimal Δ^* is optimizing the problem

$$\Delta^* = \underset{\Delta \in \Omega_\Delta}{\arg\min} - \sum_{t=1}^{T} \sum_{i=1}^{K_y} \Pi^{i}_{yt} \ln(\Delta \Pi_{xt})_i,$$

without using the Jensen inequality. As $-\ln(\Delta\Pi_{xt})$ is a convex function, the problem can be solved numerically using the Spectral Projected Gradient method[16].

We use both approaches (the analytical solution using Jensen inequality and the numerical solution) in our experiments.

3.2 The Support Vector Machine

The Support Vector Machine (SVM) is a supervised learning classifier originally designed for binary classifications. It was introduced by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963 [17].

Let $T := \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), ..., (\boldsymbol{x_n}, y_n)\}$, be the training dataset, where n is the number of the samples, $\boldsymbol{x_i} \in \mathbb{R}^m$, $i \in \{1, 2, \ldots, n\}$ is the sample and $y_i \in \{-1, 1\}$ is the label related to the sample $\boldsymbol{x_i}$. The classification model is represented by the means of the hyperplane H, defined such that:

$$H : \boldsymbol{\omega}^T \boldsymbol{x} - \widetilde{b} = 0,$$

where ω is the normalized normal vector of the hyperplane H, and

$$\widetilde{b} = \frac{b}{\|\omega\|}$$

is the bias from the origin.

First, we consider linearly separable training data. The two classes of data are distinguished by two hyperplanes so that the distance between them is maximized. The region bound by these hyperplanes is called the margin. The hyperplane that lies halfway between them is called the maximum-margin hyperplane. These hyperplanes are described by the following equation:

$$\boldsymbol{w}^T \boldsymbol{x} - \widetilde{b} = \pm 1. \tag{3.3}$$

Sample $x_i, i = 1, \ldots, n$ belongs to the positive class, i.e. $y_i = 1, i = 1, \ldots, n$, when

$$\boldsymbol{w}^T \boldsymbol{x_i} - \widetilde{b} \geq 1, \tag{3.4}$$

and the negative class, i.e. $y_i = -1, i = 1, \ldots, n$, when

$$\boldsymbol{w}^T \boldsymbol{x_i} - \widetilde{b} \leq -1. \tag{3.5}$$

The properties (3.4) and (3.5) can be combined into a single equation

$$y_i(\boldsymbol{w}^T \boldsymbol{x} - \widetilde{b}) \geq 1.$$

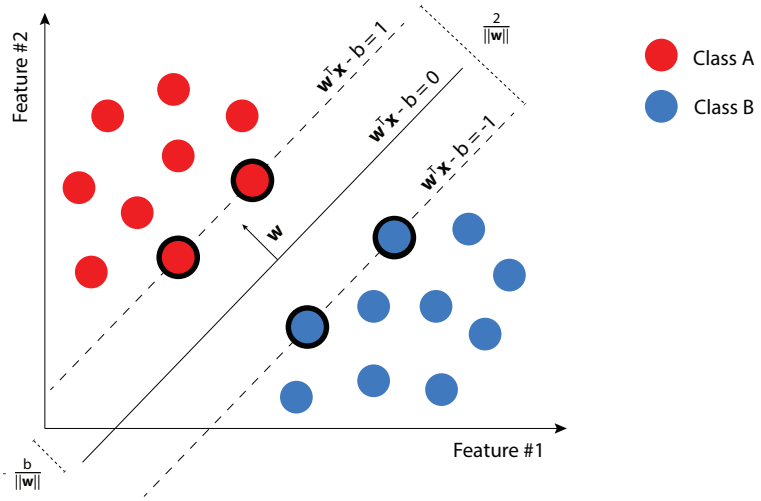


Figure 3.1: Example of the SVM model. [18]

From the Figure 3.1, we can see, the distance between hyperplanes (3.3) is $\frac{2}{\|\boldsymbol{w}\|}$. As we want to maximize this distance, we need to minimize $\|\boldsymbol{w}\|$. This leads to an optimization problem

$$\arg\min_{\boldsymbol{w},b} \|\boldsymbol{w}\| \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T \boldsymbol{x}_i - b) \geq 1, \\ i = 1, ..., n, \end{cases}$$

which can be reformulated as the Quadratic Programming problem

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T \boldsymbol{x}_i - b) \geq 1, \\ i = 1, ..., n. \end{cases} \tag{3.6}$$

Because the training data is nearly never linearly separable, the soft margin version of the SVM was proposed by Vladimir N. Vapnik and Corinna Cortes [17] in 1995. It exploits an additional function called the hinge loss function:

$$\xi_i = \max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i - b)). \tag{3.7}$$

The hinge loss function (3.7) equals 0 for a sample on the correct side of the corresponding hyperplane (3.3). However, for a sample on the wrong side of the corresponding hyperplane (3.3), the value of the function is proportional to the distance from the hyperplane.

If we add the hinge loss function (3.7) to optimization problem (3.6), we get a soft margin SVM optimization problem

$$\underset{\boldsymbol{w},b,\xi_i}{\arg\min}\ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1, ..., n, \end{cases} \tag{3.8}$$

where C is a penalty for the misclassification error. The formulation (3.8) is called the $l1$-loss $l2$-regularized SVM. The primal formulation (3.8) can be modified using the Lagrange duality with Lagrange multipliers $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_n]^\top$, $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_n]^\top$. Exploiting Karush-Kuhn-Tucker conditions, we obtain the dual formulation

$$\underset{\boldsymbol{\alpha}}{\arg\min}\ \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Y}^T\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{e} \text{ s.t. } \begin{cases} \boldsymbol{o} \leq \boldsymbol{\alpha} \leq C\boldsymbol{e}, \\ \boldsymbol{B_e}\boldsymbol{\alpha} = 0, \end{cases} \tag{3.9}$$

where $\boldsymbol{e} = [1, 1, \ldots, 1]^\top$, $\boldsymbol{o} = [0, 0, \ldots, 0]^\top$, $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}]$, $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^\top$, $Y = \text{diag}(\boldsymbol{y})$, $\boldsymbol{B_e} = [\boldsymbol{y}^T]$ and $\boldsymbol{K} \coloneqq \boldsymbol{X}^T\boldsymbol{X}$ is the Gram matrix which is symmetric positive semi-definite (SPSD)[19]. The Hessian matrix in (3.9)

$$\boldsymbol{H} \coloneqq \boldsymbol{Y}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{Y}$$

is also an SPSD matrix.

To recover the normal vector, the formula

$$\boldsymbol{w} = \boldsymbol{X}\boldsymbol{Y}\boldsymbol{\alpha}.$$

is used. The bias b can be recovered as

$$b = \boldsymbol{w} \cdot \bar{\boldsymbol{x}} - y_i,$$

where $\bar{\boldsymbol{x}}$ is the mean of all support vectors.

Instead of the linear sum of the loss functions ξ_i in (3.8), we can use the square sum of the loss functions in the objective function

$$\underset{\boldsymbol{w},b,\xi_i}{\arg\min}\ \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1 - \xi_i, \\ i = 1, ..., n. \end{cases} \tag{3.10}$$

The problem (3.10) is called $l2$-loss $l2$-regularized SVM. The dual formulation can again be obtained

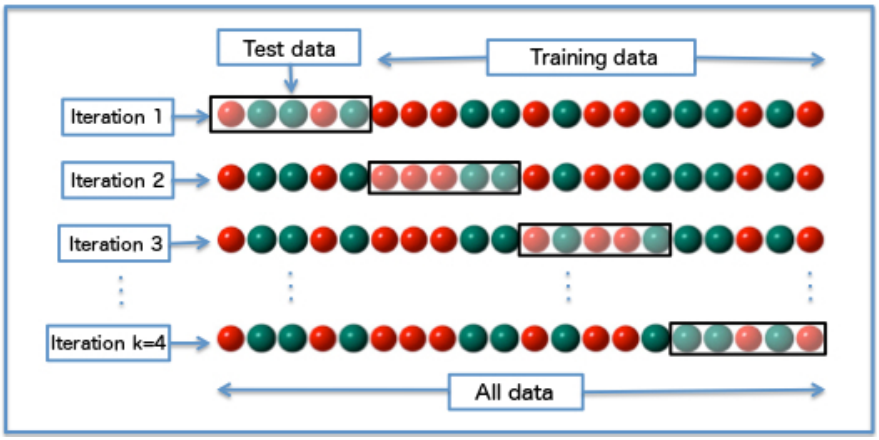


Figure 3.2: k-fold CV splits data into k folds and trains the model k times always leaving out a different fold as a validation set. [20]

by using the Lagrange duality:

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{H} + C^{-1}\boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{\alpha}^T e \text{ s.t. } \begin{cases} \boldsymbol{o} \leq \boldsymbol{\alpha}, \\ \boldsymbol{B_e}\boldsymbol{\alpha} = 0. \end{cases}$$

The Hessian matrix $\boldsymbol{H}$, regularized by the matrix $C^{-1}\boldsymbol{I}$ is symmetric positive definite, therefore, this optimization problem should be more stable than the $l1$-loss $l2$-regularized SVM problem.

3.2.1 Hyperparameter optimization

Hyperparameter optimization is the process of selecting good parameters for a classifier. In SVM, we need to find a good penalty C. One approach to hyperparameter optimization is a Grid search.

Grid search is an exhaustive searching through a user-specified subset of parameters. The classifier is trained with each combination of the subset. The combination, which yields the best result is then selected. By the best result, we mean the highest score of a user-specified metric.

To test the hyperparameter selection, we need new, independent data from the data used in a training step. This is accomplished by splitting the data into subsets. The subsets are selected using a stratified cross-validation technique.

3.2.2 Cross-validation

Cross-validation is an approach to split a dataset into a training and validation subset. For each set of hyperparameters, a dataset is split into k subsets (folds) of an equal size. One of the subsets is kept as a validation set, while the model is trained on the other $k-1$ folds. The process is repeated k times, selecting each fold as a validation subset. The results of all k runs are averaged for a final score. This process can be seen in Figure 3.2.

As a randomly selected fold might not represent the classes in the same ratio as the whole set, stratified cross-validation is used. The random folds are selected in a way, where the ratio of the classes is roughly equal to the ratio of classes in the whole dataset.

3.3 Metrics

To assess the quality of the classification model, we need to analyze several metrics. Many of such metrics are derived from a confusion matrix (see Table 3.1). The confusion matrix is generated by

	Predicted negative	Predicted positive
Actual negative	True negatives (TN)	False positives (FP)
Actual positive	False negatives (FN)	True positives (TP)

Table 3.1: Confusion Matrix

counting the testing data, which are supposed to belong to a negative class (Actual negative) or a positive class (Actual positive), and their predicted class (Predicted negative/Predicted positive).

We are considering the following metrics in our experiments:

Accuracy: How often is the classifier correct overall. This metric is represented in percentages.

$$\frac{TN + TP}{TN + FP + FN + TP}$$

Precision: How often is the classifier correct when it predicts positive.

$$\frac{TP}{TP + FP}$$

Sensitivity (Recall): How often is the classifier correct when it is actually positive.

$$\frac{TP}{TP + FN}$$

F_1 **Score:** A harmonic mean of precision and sensitivity.

$$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Generally, for each of these metrics, the higher value we get, the better the classification model.

Chapter 4

Datasets

In this chapter, we take a look at the datasets we use, to test our pipeline.

Chapter 5

Results

In this chapter, we take a look at the different results of our classification.

Chapter 6

Conclusion

In conclusion, nothing works.

Bibliography

1. LEE, Yongjin; LEE, Kyunghee; PAN, Sungbum. Local and global feature extraction for face recognition. In: *International Conference on Audio-and Video-Based Biometric Person Authentication*. 2005, pp. 219–228.
2. LOWE, David G. Object recognition from local scale-invariant features. In: *Proceedings of the seventh IEEE international conference on computer vision*. 1999, vol. 2, pp. 1150–1157.
3. CHRIS HARRIS, Mike Stephens. A COMBINED CORNER AND EDGE DETECTOR. *Plessey Research Roke Manor*. 1988.
4. CANNY, John F. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986.
5. KOENDERINK, Jan J. The structure of images. *Biological cybernetics*. 1984, vol. 50, no. 5, pp. 363–370.
6. MIKOLAJCZYK, Krystian. *Detection of local features invariant to affines transformations*. 2002. PhD thesis. Institut National Polytechnique de Grenoble - INPG.
7. LOWE, David G. Distinctive Image Featuresfrom Scale-Invariant Keypoints. *International Journal of Computer Vision*. 2004.
8. BRADSKI, Gary. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000.
9. BAY, Herbert; TUYTELAARS, Tinne; VAN GOOL, Luc. Surf: Speeded up robust features. In: *European conference on computer vision*. 2006, pp. 404–417.
10. RUBLEE, Ethan; RABAUD, Vincent; KONOLIGE, Kurt; BRADSKI, Gary. ORB: An efficient alternative to SIFT or SURF. In: *2011 International conference on computer vision*. 2011, pp. 2564–2571.
11. ROSTEN, Edward; DRUMMOND, Tom. Machine learning for high-speed corner detection. In: *European conference on computer vision*. 2006, pp. 430–443.
12. CALONDER, Michael; LEPETIT, Vincent; STRECHA, Christoph; FUA, Pascal. Brief: Binary robust independent elementary features. In: *European conference on computer vision*. 2010, pp. 778–792.

13. ROSIN, Paul L. Measuring corner properties. *Computer Vision and Image Understanding*. 1999, vol. 73, no. 2, pp. 291–307.

14. FLÖCK, Fabian. *K-fold cross validation* [online]. 2016 [visited on 2021]. Available from: `https://commons.wikimedia.org/wiki/File:K-fold_cross_validation_EN.jpg`.

15. KULLBACK, Solomon; LEIBLER, Richard A. On information and sufficiency. *The annals of mathematical statistics*. 1951, vol. 22, no. 1, pp. 79–86.

16. BIRGIN, Ernesto G; MARTINEZ, Jose Mario; RAYDAN, Marcos. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*. 2000, vol. 10, no. 4, pp. 1196–1211.

17. CORINNA CORTES, Vladimir Vapnik. Support-vector networks. *Machine Learning*. 1995-09, vol. 20, no. 3, pp. 273–297. ISSN 1573-0565. Available from DOI: `10.1007/BF00994018`.

18. KRUŽÍK, Jakub; PECHA, Marek; HAPLA, Václav; HORÁK, David; ČERMÁK, Martin. Investigating Convergence of Linear SVM Implemented in PermonSVM Employing MPRGP Algorithm. In: KOZUBEK, Tomáš; ČERMÁK, Martin; TICHÝ, Petr; BLAHETA, Radim; ŠÍSTEK, Jakub; LUKÁŠ, Dalibor; JAROŠ, Jiří (eds.). *High Performance Computing in Science and Engineering*. Cham: Springer International Publishing, 2018, pp. 115–129. ISBN 978-3-319-97136-0.

19. MAREK PECHA, David Horák. Analyzing l1-loss and l2-loss Support Vector Machines Implemented in PERMON Toolbox. In: *Bioanalytical Reviews*. Springer Berlin Heidelberg, 2018-04, pp. 13–23. Available from DOI: `10.1007/978-3-030-14907-9_2`.

20. FLÖCK, Fabian. *K-fold cross validation* [online]. 2016 [visited on 2021]. Available from: `https://commons.wikimedia.org/wiki/File:K-fold_cross_validation_EN.jpg`.