UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

Data Science and Business Informatics

Project 25
Multi-class SVM with Interior-Point

Professor:

Prof. Antonio Frangioni

Students:

Pietro Argento
Minh Duc Pham

ANNO ACCADEMICO 2023/2024

Contents

Chapter 1

Introduction

1.1 Description of the task

This report explores two different optimization algorithms applied to a ML model.

The ML Model is a Support Vector Machine-type approach with linear kernel for multi-class classification, using the standard One-to-Rest approach.

The first Optimization Algorithm is an algorithm of the class of interior-point methods applied to the dual formulation of the SVC.

The second optimization algorithm is a general-purpose solver applied to an appropriate formulation of the problem.

No off-the-shelf solver will be used, save of course for the second algorithm. Specific discussion will be provided about if and how solving one training problem in the One-to-Rest approach may provide useful solution that may help in solving the others.

Chapter 2

SVM Problem Formulation

The goal of this chapter is to describe the Support Vector Machine and use the Lagrangian dual formulation in order to define the optimization task as a quadratic programming problem.

2.1 Hard margin formulation

The Support Vector Machine model use an hyperplane H to compute the predictions with the aim of classifying the data in the correct class. In particular, H is defined by a vector of parameters $w = w_1, w_2, ...$ and a bias b. In fact, it can be written as

$$H(x) = \mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + ... + b$$

.

Note that $\mathbf{y}$ is the vector that contains the labels corresponding to the class of each point. In the simplest formulation of the SVM, the points can be classified only in two different classes, represented by the label $+1$ for one class and -1 for the other. As a consequence, the vector $\mathbf{y}$ containing the actual classes is

$$\mathbf{y}^T = [\pm 1, \pm 1, ...] \Rightarrow y_i = \pm 1$$

.

The margin M is the distance from the closest point x_i of the dataset to the hyperplane H. It can be computed as

$$M(x_i) = \frac{|\mathbf{w}^T\mathbf{x} + b|}{||\mathbf{w}||}$$

The support vectors $\mathbf{x}_{sv}$ are exactly those points on the margin, in other words the points (or point) closest to the hyperplane. To simplify the problem, it is possible to define a new hyperplane $\bar{H}$, where the new parameters with respect to the previous ones are $\bar{\mathbf{w}} = \frac{\mathbf{w}}{H(\mathbf{x}_{sv})}$ and $\bar{b} = \frac{b}{H(\mathbf{x}_{sv})}$. This allows to set the value of $\bar{H}(\mathbf{x}_{sv}) = 1$. Note that these steps can be done WLOG (Without loss of generality).

Now, the margin can be rewritten as

$$M(\mathbf{x}_{sv}) = \frac{|\bar{\mathbf{w}}^T x + \bar{b}|}{||\bar{\mathbf{w}}||} = \frac{1}{||\bar{\mathbf{w}}||}$$

.

The idea of the SVM model is to find the hyperplane H that maximizes the margin M. The objective can function of the optimization problem therefore is

$$\max_{\mathbf{w},b} M(\mathbf{x}_{sv}) \Rightarrow \max_{\mathbf{w},b} \frac{1}{||\mathbf{w}||} \Rightarrow \min_{\mathbf{w},b} ||w|| \Rightarrow \min_{\mathbf{w},b} ||\mathbf{w}||^2$$

Now the constraint. In this basic formulation, misclassification is not allowed, for this reason, the actual class y and the predicted class $\bar{y}$ using the hyperplane $H(x) = \mathbf{w}^T\mathbf{x} + b$ must have the same sign. Note that $y_i = \pm 1$ and $H(x) \geq 1$, because H was designed to have a minimum value of 1 given by the support vectors. The constraint is then

$$y \cdot \bar{y} \geq 1 \Rightarrow y \cdot (H(\mathbf{x})) \geq 1 \Rightarrow y \cdot (\mathbf{w}^T\mathbf{x} + b) \geq 1$$

Together, the objective function and the constraint of what is called "Hard Margin SVM" is

$$\begin{aligned} \min_{\mathbf{w},b} \quad & ||\mathbf{w}||^2 \\ \text{s.t.} \quad & y_i \cdot (\mathbf{w}^T x_i + b) \geq 1 \quad \forall i \in \{1, ..., n\} \end{aligned}$$

The vector of weights $\mathbf{w}$ and the scalar b solving the problem are the values that will be used in the classifier $\mathbf{w}^T\mathbf{x} + b$

2.2 Soft margin formulation

In the "Hard Margin SVM" just described, misclassification was not allowed. However, in real-world dataset, it can happen and it is important to allow a certain amount of it. The first change must be made in the constraint

$$y_i \cdot (w^T x_i + b) \geq 1 - \xi_i$$

The $\xi_i \geq 0$ serves as a threshold to allow misclassification, because now y and $\bar{y}$ can have different sign. Note that there is a different ξ_i for each point. However, ξ must be as small as possible and to reach this goal it is sufficient to add $\sum \xi_i$ in the objective function which is being minimized. Furthermore, a constant value C can be added to change the impact of ξ into the objective function. The final formulation of the "Soft Margin SVM" is

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & ||w||^2 + C \sum_{i}^{N} \xi_i \\ \text{s.t.} \quad & y_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

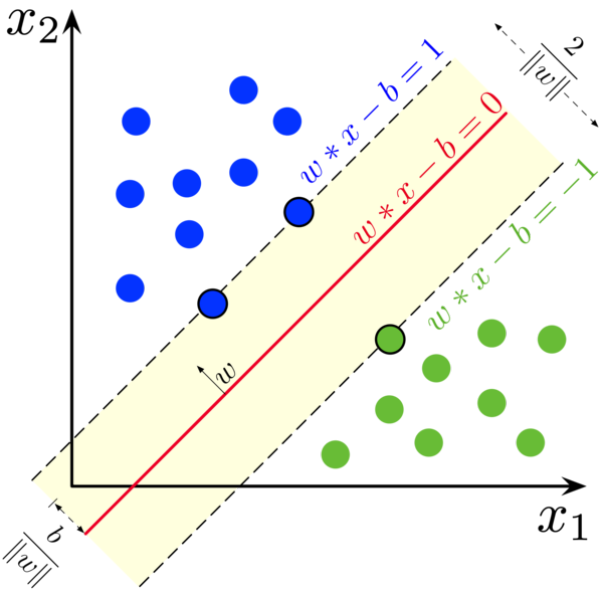

Figure 2.1: Hard Margin SVM from Wikipedia

Or, more often, expressed as follows, including $\frac{1}{2}$ for simplifing the gradient.

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i}^{N}\xi_i \\ \text{s.t.} \quad & y_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{2.1}$$

2.3 Dual formulation

For a convex optimization problem with inequality constraints

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \quad i = 1, ..., m \\ & Ax = b\,, \end{aligned}$$

the Lagrangian Dual problem is

$$\begin{aligned} \max_{\lambda} \quad & g(\lambda, \nu) \\ \text{s.t.} \quad & \lambda_i \geq 0 \quad i = 1, ..., m \end{aligned} \tag{2.2}$$

.

The function $g(\lambda, \nu)$ is called Lagrange Dual function and is defined as

$$\begin{aligned} g(\lambda, \nu) &= \inf_x \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_i (Ax - b) \right) \\ &= \inf_x \mathcal{L}(x, \lambda, \nu)\,. \end{aligned} \tag{2.3}$$

Since (2.3) can be used as a lower bound for (2.1)(lower bound property), we aim at maximizing it through the Dual Problem (2.2). Furthermore, if all the conditions for strong duality holds, a optimal solution of the Dual Problem is also a optimal solution of the Primal Problem. The goal of this section is to derive the Lagrange Dual function in order to write the Lagrange Dual problem (2.2) that will be used in the solver. Then, the (2.1) variables will be obtained from dual variables.

The KKT conditions of a generic convex problem are

$$\begin{aligned} \nabla\mathcal{L}(x,\lambda,\nu) &= 0 && \text{gradient condition} \\ f_i(x) &\leq 0 && \text{primal feasibility} \\ \lambda_i &\geq 0 && \text{dual feasibility} \\ \lambda_i f_i(x) &= 0 && \text{complementarity condition} \end{aligned}$$

Back to the primal formulation of the SVM problem, the Lagrangian $\mathcal{L}$ can be written as

$$\begin{aligned} \mathcal{L}(\mathbf{w},b,\xi,\alpha,\beta) =& \frac{1}{2}||\mathbf{w}||^2 + C\sum_i^m \xi_i \\ &+ \sum_i^m \alpha_i \Big(- y_i(\mathbf{w}^T\mathbf{x}_i + b) + (1-\xi_i)\Big) + \sum_i^m \beta_i(-\xi_i), \end{aligned}$$

note that $\alpha_i \geq 0$ and $\beta_i \geq 0$ are the Lagrangian multipliers.

With the appropriate rearrangements, we write the equalities of the KKT conditions:

$$\mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = 0 \qquad \text{gradient condition } (\mathbf{w}) \tag{2.4}$$

$$-\sum_{i=1}^n \alpha_i y_i = 0 \qquad \text{gradient condition } (b) \tag{2.5}$$

$$C - \alpha_i - \beta_i = 0 \qquad \text{gradient condition } (\xi_i \,\forall i) \tag{2.6}$$

$$\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - (1-\xi_i)) = 0 \qquad \text{complementarity cond. } (\alpha_i \,\forall i) \tag{2.7}$$

$$\beta_i\xi_i = 0 \qquad \text{complementarity cond. } (\beta \,\forall i) \tag{2.8}$$

We can use (2.4) and (2.6) into the Lagrangian $\mathcal{L}$ as follows:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_i^n \xi_i \\
&\quad + \sum_i^n \alpha_i \Big(- y_i(\mathbf{w}^T\mathbf{x}_i + b) + (1-\xi)\Big) + \sum_i^m \beta_i(-\xi_i) \\
&= \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j - \sum_i \alpha_i y_i \mathbf{w}^T\mathbf{x}_i \\
&\quad - b\sum_i \alpha_i y_i + \sum_i \alpha_i + C\sum_i \xi_i - \sum_i \alpha_i\xi_i - \sum_i \beta_i\xi_i \qquad (2.9) \\
&= \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j - \sum_i \alpha_i y_i(\alpha_j y_j \mathbf{x}_j)\mathbf{x}_i - 0 + \sum_i \alpha_i + 0 \\
&= \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j - \sum_i \alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j + \sum_i \alpha_i \\
&= -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j + \sum_i \alpha_i
\end{aligned}$$

For the sake of completeness, we write also the inequalities of the KKT for primal and dual feasibility.

$$\begin{aligned}
y_i \cdot (\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i \geq 0 \qquad & \text{primal feasibility (misclass)} \\
\xi_i \geq 0 \qquad & \text{primal feasibility } (\xi) \\
\alpha_i \geq 0 \qquad & \text{dual feasibility } (\alpha) \qquad (2.10) \\
\beta_i \geq 0 \qquad & \text{dual feasibility } (\beta) \qquad (2.11)
\end{aligned}$$

Writing (2.6) as $\beta_i = C - \alpha_i$ with $\beta_i \geq 0$, we can replace (2.11) with $\alpha_i \leq C$. Together with (2.10), we will use the constraint $0 \leq \alpha_i \leq C$.
Observe that the stationarity constraint (2.5) was not used in the calculations for the new objective function, hence it will be added to the Dual Problem as well.

Now, we can write the Dual problem for the SVM problem as $\max_\alpha \mathcal{L}(\alpha)$

$$\begin{aligned}
\max_\alpha \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \\
\text{s.t.} \quad & \sum_{i=1}^n y_i\alpha_i = 0 \\
& 0 \leq \alpha_i \leq C.
\end{aligned} \qquad (2.12)$$

Although the dual originates from a maximization problem, in practice we will write it as a minimization problem of the negative dual function.

2.4 Primal and Dual variables

Once the dual problem is solved and α^* is found, how to build back the primal variables $(\mathbf{w}^*, b^*, \xi^*)$?
First, $\mathbf{w}^*$ can be obtained from the gradient condition (2.4) as

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

Second, b^*, that is a scalar, needs some reasoning. If we look at the complementary slackness (2.7), we observe that:

- if $\alpha_i = 0$, the constraint (2.7) is inactive;
- if $\alpha_i > 0$, the constraint (2.7) is active; this happens when $y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) = 1 - \xi_i$;
- if $\alpha_i > 0$ and $\alpha_i < C$, the constraint (2.7) is active and, from (2.6) we have $\beta_i > 0$. This forces also the other complementarity constraint (2.8) to be active, meaning $\xi_i = 0$, and now we can write (2.7) as $y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) = 1$.

From the last point, we derive the following equation valid for points where $0 < \alpha_i < C$, given $(y_i)^2 = 1$ with $y_i = \pm 1$

$$\begin{aligned} y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) &= 1 \\ y_i \cdot \mathbf{w}^T \mathbf{x}_i + y_i \cdot b &= 1 \\ \mathbf{w}^T \mathbf{x}_i + b &= y_i \\ b &= y_i - \mathbf{w}^T \mathbf{x}_i. \end{aligned}$$

It is important to clarify that these values for the primal variables are valid only if the KKT are respected, i.e. optimality is reached.

There is actually a smarter, and much more precise, idea to obtain b. Since the path just described gets only an approximate value for b, the resulting primal-dual gap will suffer due to numerical reconstruction of primal variables, in particular the mean for b. Alternatively, we can observe from (2.9) that the role of b is exactly the role of the scalar Lagrangian variable for the equality constraint $\sum_{i=1}^{n} y_i \alpha_i = 0$ in (2.12). If this Lagrangian multiplier, that in the following sections will be called ν, is available, like in primal-dual solvers described in this report where ν is computed directly, it can be reused as b without estimating it from support vectors, i.e.

$$b = \nu$$

Finally, ξ can be recovered from (2.8) with

$$\xi_i = \max(0, 1 - (\mathbf{w}^T \mathbf{x}_i + b)).$$

2.5 Kernel trick

An important upgrade in SVM classifiers is the use of kernels in the dual formulation. In particular, in the objective function, the dot product $x_i x_j$, which results in a linear classifier, can be replaced with a nonlinear kernel function $K(x_i x_j)$.

The result is a more general formulation of the dual:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(x_i x_j) - \sum_i \alpha_i \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{n} y_i \alpha_i = 0 \end{aligned}$$

The four basic kernels are:

- **linear**: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.
- **polynomial**: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d$, $\gamma > 0$.
- **radial basis function (RBF)**: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, $\gamma > 0$.
- **sigmoid**: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$.

Here, γ, r, and d are kernel parameters. This report will be focused on the linear kernel.

Chapter 3

Feasible Primal-Dual SVM

3.1 Interior point methods

In this chapter, we start by introducing Interior Point Methods (IPM), a class of algorithms fundamentally distinct from traditional approaches. It was introduced by Karmarkar in 1981, IPMs were later developed into a comprehensive framework by Nesterov and Nemirovski in 1994.

Convex optimization problems that we aim to solve with the *interior-point methods* are in the form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b, \end{array} \tag{3.1}$$

where $f_0, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ are convex and twice continuously differentiable, and $A \in \mathbb{R}^{p \times n}$ with rank $A = p < n$. We assume that the problem is solvable, i.e., an optimal x^* exists and we denote the optimal value $f_0(x^*)$ as p^*.

We also assume that there exists $x \in \mathcal{D}$ that satisfies $Ax = b$ and $f_i(x) < 0$ for $i = 1, \ldots, m$, in other words, the problem is strictly feasible. This means that Slater's condition holds, therefore strong duality holds. So there exist dual optimal $\lambda^* \in \mathbb{R}^m$, $\nu^* \in \mathbb{R}^p$, which together with x^* satisfy the KKT conditions. [1, §11.1]

$$\begin{array}{ll} Ax^* = b, f_i(x^*) & \leq 0, \quad i = 1, \ldots, m \\ \lambda_i^* & \geq 0 \\ \nabla f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + A^T \nu^* & = 0 \\ \lambda_i^* f_i(x^*) & = 0, \quad i = 1, \ldots, m. \end{array}$$

Interior-point methods solve the problem 3.1 (or the KKT conditions (3.1) by applying Newton's method to a sequence of equality constrained problems, or to a sequence of modified versions of the KKT conditions.

In order to solve the IPM problems, there are several methodologies, but primarily including the Barrier Method and the Primal-Dual Method.

- Barrier Method employs logarithmic barrier functions to manage inequality constraints, transforming constrained optimization problems into unconstrained problems.
- Primal-Dual Method optimizes both the primal and dual formulations of the problem, facilitating a more comprehensive solution strategy. These foundational approaches leverage techniques such as duality theory and Newton's method to ensure efficient convergence toward optimal solutions.

For this report, we will concentrate our efforts on the *primal-dual method*.

3.2 Primal-Dual Method for QP

The dual formulation of SVM problem can be seen as a Quadratic Problem (QP) with an equality constraint and a box-constraint. For this reason, in this chapter we will explore how to deal specifically with this type of problem.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Qx + qx \\ \text{subject to} & Ax = b \\ & 0 \leq x_i \leq u_i, \quad i = 1, \ldots, m, \end{array} \tag{3.2}$$

Primal-dual interior-point methods are a class of algorithms used to solve convex optimization problems, closely related to barrier methods but with key differences that often make them more efficient in practice.

Unlike the barrier method, which distinguishes between inner and outer iterations, primal-dual methods perform all updates within a single iteration loop, simultaneously updating both primal and dual variables.

The search directions in primal-dual methods are derived from Newton's method applied to the perturbed Karush-Kuhn-Tucker (KKT) conditions, tailored to the logarithmic barrier formulation.

An important characteristic of primal-dual interior-point methods is that iterates need not remain feasible throughout the optimization process.

In Algorithm 1, we present the framework of primal-dual interior-point method for to the standard convex optimization setting. In the following sections, a tailored vesion for the SVM problem will be expored.

3.3 Primal-dual Method for Dual SVM

The dual SVM optimization problem can be expressed as a *Quadratic Programming (QP)* problem. The objective is to find the optimal Lagrange multiplier α. We use the formula (2.12):

Algorithm 1 Primal-Dual Interior-Point Method

1: Initialize primal and dual variables, and set centering parameter μ
2: **while** not converged **do**
3: Obtain search direction from perturbed KKT conditions
4: Choose step size
5: Update primal and dual variables
6: Reduce barrier parameter μ
7: **end while**

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \\ & 0 \le \alpha_i \le C, \quad \forall i = 1, \ldots, n \end{aligned}$$

In this chapter, we follow a common notation in optimization papers, replacing the dual variable α with z. This change is necessary to avoid confusion with the matrix A while using uppercase letters as diagonal matrices for variables (e.g. Z := diag(z)).

We rewrite the problem as

$$\begin{aligned} \min_{z} \quad & \tfrac{1}{2}z^T Q z - e^T z \\ \text{subject to} \quad & y^T z = 0 \\ & 0 \le z \le Ce \end{aligned}$$

.

Here, we recall that Q is the Kernel Matrix obtained from a Kernel Function (2.5); **e** is the vector of ones of dimension n; C is the hyperparameter that define the cost of misclassification.

The choices regarding Lagrangian form and multipliers' notation reflect those in [1, §5], i.e. λ_i are the Lagrange multipliers for inequalities, while ν are those related to equalities.

The Lagrangian relaxation, that will be useful in the following sections, is

$$\mathcal{L}(z, \lambda_-, \lambda_+, \nu) = \frac{1}{2}z^T Q z - e^T z + \nu^T (y^T z) + \lambda_-^T(-z) + \lambda_+^T(z - Ce) \tag{3.3}$$

.

3.4 Pseudocode

The pseudocode can be expressed as follows:

Algorithm 2 Feasible-start primal-dual interior-point method for SVM dual

1: Initialize feasible variables: $\alpha, \nu, \lambda_-, \lambda_+, \mu$ such that all constraints are satisfied

2: Initialize parameters: $\mu_0 = \frac{z^T\lambda_- + (Ce-z)^T\lambda_+}{2n}, \varepsilon_{\text{gap}} > 0, \quad \sigma \in (0,1)$

3: **while** $\text{gap}_c \geq \varepsilon_{\text{gap}}$ **do**

4: Compute surrogate gap:

$$\text{gap}_c = \lambda_-^T z + \lambda_+^T (C\mathbf{e} - z)$$

5: Solve the KKT system to get primal-dual search directions

$$\Delta z, \Delta\nu, \Delta\lambda_-, \Delta\lambda_+$$

6: Compute maximum step length $\eta_{\max}$ to preserve strict feasibility, i.e.

$$\lambda_- + \eta\Delta\lambda_- > 0, \quad \lambda_+ + \eta\Delta\lambda_+ > 0, \quad z + \eta\Delta z \in (0, C)^n$$

and set step length: $\eta \leftarrow 0.9995 \cdot \eta_{\max}$

7: Update variables:

$$\begin{aligned} z &\leftarrow z + \eta\Delta z \\ \nu &\leftarrow \nu + \eta\Delta\nu \\ \lambda_- &\leftarrow \lambda_- + \eta\Delta\lambda_- \\ \lambda_+ &\leftarrow \lambda_+ + \eta\Delta\lambda_+ \end{aligned}$$

8: Reduce central path parameter:

$$\mu \leftarrow \sigma\mu$$

9: **end while**

3.5 Parameters' Initialization

In the literature, different heuristics can be found regarding the initialization of the Interior Point parameters, i.e. μ and σ. We follow [4], which lead to the following choices.

We defined μ as the barrier parameter, more appropriately called the central path parameter. We know that, during the iterations, only the complementarity conditions are violated:

$$\lambda_-^\top \mathbf{z} = \mu e, \quad \lambda_+^\top (Ce - \mathbf{z}) \ = \mu e, \quad \text{where } \mu \to 0$$

Thus, we have the formula to calculate to update the value of μ as below, which uses the centrality gap as numerator and the number of inequalities $m = 2n$ as denominator:

$$\mu_0 = \frac{z^T \lambda_- + (Ce - z)^T \lambda_+}{2n}$$

Regarding the reduction of μ, i.e. σ different ideas might be implemented. The simplest is to use a parameter $\sigma = \frac{1}{m}$ where m is the number of inequalities. Alternatives involve Mehrotra-s Predictor-Corrector or Centrality-based updates [3]. We decided to explore in our experiments values of

$$\sigma \in [0, 1].$$

3.6 Starting Point

Compared to the general QP described before (3.2), the dual SVM problem can be easily initialised with a feasible point.

Considering the Lagrangian (3.3), we need to initialize the following dual variables:

- $\lambda_- \geq 0$ for the lower bound $\alpha_i \geq 0$,
- $\lambda_+ \geq 0$ for the upper bound $\alpha_i \leq C$,
- $\nu \in \mathbb{R}$ for the equality constraint $y^T z = 0$.

Given the primal problem, the vector $z \in \mathbb{R}^n$ is feasible for the dual SVM problem if it strictly satisfies the box constraints (inequalities): $0 \leq z_i \leq C \quad \forall i = 1, \ldots, n$

Given the Lagrangian dual, we can choose strictly positive dual variables:

$$(\lambda_-)_i = \varepsilon > 0, \quad (\lambda_-)_i = \varepsilon > 0 \quad \text{(e.g., } \varepsilon = 0.1)$$

In practice, for interior-point methods, choosing any strictly feasible $z \in (0, C)^n$ (i.e. respecting primal feasibility), and setting small positive values for λ_- and λ_+ (i.e. respecting dual feasibility), is sufficient.

However, in a feasible-start primal-dual, we want the primal and dual residuals to be zero already at the first iteration. This means an initial point that satisfies

$$y^T \alpha = 0, \quad \nabla_z \mathcal{L} = Qz - e - \nu y - \lambda_- + \lambda_+ = 0$$

.

For example, in order to satisfies both the box constraints and the equality constraint with z, our choice is:

$$z_i = \begin{cases} \frac{C}{2}, & \text{for the smaller class} \\ \text{class ratio} * \frac{C}{2}, & \text{for the larger class} \end{cases}$$

where class ratio $= \frac{\text{no instances smaller class}}{\text{no instances larger class}}$.

For what concerns the dual variables, to respect the stationarity condition, we can set for simplicity $\nu = 0$, which leads to $-\lambda_- + \lambda_+ = e - Qz$. Now, adding a δ on both sides to ensure non-negativity of dual variables gives us

$$\nu = 0, \quad (\lambda_-)_i = \max(-(e - Qz), 0) + \delta, \quad (\lambda_+)_i = \max(e - Qz, 0) + \delta$$

The value of δ should not be too small to ensure that the initial point lies well in the interior of the feasible region. In practice, we consider values like $\delta = 1$ or $\delta = 10$.

3.7 Primal-dual search direction

The following derivation of the search direction is ours, expanding the lecture slides. In this section, only relevant partial results are reported, the full derivation can be found in the Appendix 7.1. A similar approach can be found in [7].

Instead of solving the original problem, we aim at iteratively (and approximately) solving a modified problem with a barrier term connected to the inequalities. Given this new "barrier problem" (also called "**centering problem**"), rather than the original KKT system, the idea is to take one Newton step to solve a perturbed version of the KKT conditions (also called "centrality conditions [1, p. 11.2]") for (3.3), then reduce the perturbation and iteratively solve again until convergence. More about the central path can be found in the Appendix 7.3.

Therefore, we now consider the **optimality conditions of the centering problem**. Inequalities will be considered when taking care of the stepsize, on the other hand, to calculate the step direction, we focus on equalities writing the

matrix form of the perturbed KKT. Since we do not aim at an exact solution of the system, for computing a Newton step, we will use a **linearized version of the KKT** removing the bilinear terms $\Delta\lambda_-\Delta z$ and $\Delta\lambda_+\Delta z$. In the following system and in later steps, note that $U := diag(Ce)$ and $A = y^T$:

$$\begin{bmatrix} Q & A^T & -I & I \\ A & 0 & 0 & 0 \\ \Lambda_- & 0 & Z & 0 \\ -\Lambda_+ & 0 & 0 & U-Z \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\nu \\ \Delta\lambda_- \\ \Delta\lambda_+ \end{bmatrix} = \begin{bmatrix} r_z \\ r_{nu} \\ r_{lm} \\ r_{lp} \end{bmatrix} \tag{3.4}$$

Given that we removed the bilinear terms, we have:

$$r_z = -(Qz - e - \lambda_- + \lambda_+ + y^T\nu) \tag{3.5}$$
$$r_\nu = -(y^T z) \tag{3.6}$$
$$r_{lm} = -(\lambda_- z - \mu e)$$
$$r_{lp} = -(\lambda_+(U - z) - \mu e).$$

After some calculations, we can build the **augmented system** as

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} r_c \\ r_\nu \end{bmatrix},$$

where

$$H = Q + Z^{-1}\Lambda_- + (U - Z)^{-1}\Lambda_+$$
$$A = y^T$$
$$r_c = -(Qz - e + y^T\nu - \mu Z^{-1}e + \mu(U - Z)^{-1}e)$$
$$r_\nu = -(y^T z)$$

We observe that the term $H - Q$ is diagonal and strictly positive, hence H is strictly positive definite and nonsingular.

Also, we can work on r_c using some considerations. In the central path, we expect the stationarity constraint to be respected (i.e. $\nabla\mathcal{L}(z, \lambda_-, \lambda_+, \nu) = 0$, meaning $Qz - e + y^T\nu = +\lambda_- - \lambda_+$. Hence, we can rewrite r_c as

$$r_c = -\lambda_- + \lambda_+ + \gamma$$

Furthermore, the precomputed $\gamma = \mu Z^{-1}e - \mu(U - Z)^{-1}e$ just introduced in the steps above, we be written in a different form that can greatly improve numerical stability,

$$\gamma_i = \mu \cdot \frac{u_i - 2z_i}{z \cdot (u_i - z_i)} \tag{3.7}$$

.

The augmented system can be further simplified obtaining the **normal equation**

$$M\Delta\nu = r_b\,,$$

eliminating Δz through block elimination, where emerged the Schur's Complement $M := AH^{-1}A^T$,.

In some papers, such as [7], the matrix H is written as $Q + \Theta^{-1}$, given $\Theta^{-1} := Z^{-1}\Lambda_- + (U - Z)^{-1}\Lambda_+$, which leads to write $M := A(Q + \Theta^{-1})^{-1}A^T$.

Considering the stable form of γ (3.7), we can write the terms of the normal equation $M\Delta\nu = r_b$ as

$$\begin{aligned} M &= AH^{-1}A^T \\ r_b &= AH^{-1}r_c + A^T z \end{aligned}$$

Once $\Delta\nu$ is computed, the **other deltas** can be obtained looking back at (3.7)

$$\begin{aligned} \Delta z &= H^{-1}(r_c - A^T\Delta\nu) \\ \Delta\lambda_- &= \mu Z^{-1}e - Z^{-1}\Lambda_-\Delta z - \lambda_- \\ \Delta\lambda_+ &= \mu(U - Z)^{-1}e + (U - Z)^{-1}\Lambda_+\Delta z - \lambda_+ \end{aligned}$$

3.8 Stepsize Choice

In the feasible-start primal-dual method, the iterates always satisfy both the primal and dual feasibility conditions. They can be recovered from the full KKT conditions available in the Appendix 7.1.

As a consequence, the **feasible stepsize** can be computed as

$$\begin{aligned} \eta_{\max} := \min\{1,\ &\min_{i:\Delta(\lambda_-)_i<0}\left(-\frac{(\lambda_-)_i}{\Delta(\lambda_-)_i}\right), \min_{i:\Delta(\lambda_+)_i<0}\left(-\frac{(\lambda_+)_i}{\Delta(\lambda_+)_i}\right), \\ &\min_{i:\Delta z_i<0}\left(-\frac{z_i}{\Delta z_i}\right), \min_{i:\Delta z_i>0}\left(\frac{C - z_i}{\Delta z_i}\right)\} \end{aligned}$$

Then, in order to remain within the interior of the feasible region, we use a common **reduction factor** with the additional step

$$\eta \leftarrow 0.9995 \cdot \eta_{\max}$$

This section is radically different in infeasible-start methods (e.g. [1, §11.7]), where the iterates may violate constraints and residuals are used to declare convergence. See also the Chapter on `quadprog`.

3.9 Convergence condition

There are at least three different gaps that could be used in the algorithm. They are described in [1, §11.7.2][3].

The first is the **true primal-dual gap**, given the objective functions

$$v := \frac{1}{2}\alpha^T Q\alpha - e^T\alpha, \quad p := \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i,$$

It reflects optimality independently of the barrier parameter μ:

$$\text{gap}_p := p - (-d),$$

where v and p are dual and primal objective values, respectively. Observe that we use the dual with a negative sign because both the primal and dual formulations are treated as minimization problems. Then, at optimum there values have simply opposite signs but (approximately) same value, since strong duality holds. However, we observe that obtaining the value p is quite expensive because it requires the original data to recover the value od w. For this reason, we explore now more efficient surrogate gaps that can be used.

Since we start from a feasible point and preserve feasibility at every iteration, the primal residual (3.6) and dual residual (3.5) remain zero. Nevertheless, the only gap from solving the centering problem exactly is the complementarity constraint, thus we define the **complementarity gap** as

$$gap_c = \lambda_-^T\alpha + \lambda_+^T(C\mathbf{e} - \alpha)$$

Finally, if $x^*(\mu)$ is the exact solution of the centering problem, another **surrogate gap** can be obtained simply as

$$gap_\mu = m \cdot \mu\,.$$

The reason behind this expression can be found in the Appendix 7.3. However, since in our implementation we reduce μ manually with the reduction factor σ, it is not meaningful to choose gap_μ as the flag for convergence.

Furthermore, since we do not aim at an exact centering step, it is not safe to use it, on the other hand, it can still be seen as a target and used to tune the hyperparameter μ. In order to understand how well the centering step is computed, and hence how close we are to the central path, we might look at residuals of (3.4). In particular, when the residuals are close to zero, we expect $gap_c \approx gap_\mu$. In practice, they cannot be the same, since the KKT system is linearized and an exact solution for the problem cannot be achieved. Comparing this value gap_μ with the actual primal-dual gap gap_p or with gap_c may help evaluate whether μ is being reduced appropriately. For instance, the reduction factor σ could be adaptively adjusted, decreasing it only when gap_μ is reasonably close to $d - p$.

Our choice is to use gap_c for a given tolerance tol, e.g., 10^{-8}. It is the best compromise in terms of reliability and efficiency. A much more detailed explanation on the three gaps can be found in the Appendix 7.3.

Chapter 4

The quadprog Solver in MATLAB

4.1 Mapping the SVM Dual to a Generic QP

As shown in Chapter 3, the dual SVM training problem is a special case of a quadratic program (QP). Consequently, any off-the-shelf QP solver can be used to obtain the optimal Lagrange multipliers $\boldsymbol{\alpha}$. We selectedMATLAB's `quadprog`.

According to the documentation [6]: `quadprog` is a solver for quadratic objective functions with linear constraints. It finds a minimum for a problem specified by

$$\min_x \ \frac{1}{2}x^\top Hx + f^\top x \quad \text{such that} \quad \begin{cases} Ax \leq b, \\ A_{\text{eq}}x = b_{\text{eq}}, \\ \text{lb} \leq x \leq \text{ub}, \end{cases} \tag{4.1}$$

where H, A, and A_{eq} are matrices, and f, b, b_{eq}, lb, ub, and x are vectors.

For the soft-margin SVM, the dual takes the form

$$\min_\alpha \ \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \quad \text{such that} \quad \begin{cases} y^T\alpha = 0 \\ 0 \leq \alpha \leq C\mathbf{e}, \end{cases} \tag{4.2}$$

where $Q = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{e}$ is the all-ones vector.

Mapping (4.1) onto (4.2) is therefore straightforward:

SVM variable	`quadprog` parameter
Q	`H`
$-\mathbf{e}$	`f`
none	`A`, `b`
$\mathbf{y}^\top$	`Aeq`
0	`beq`
$\mathbf{0}$, $C\mathbf{e}$	`lb`, `ub`

4.2 The `interior-point-convex` Algorithm

`quadprog`'s default engine for convex problems is `interior-point-convex`. Its internal workflow can be summarised as follows:

1. **Presolve / Postsolve**. Eliminates redundancies, detects bound inconsistencies, and scales the problem.
2. **Initial-point generation**. Starts from $\mathbf{x}0 = \mathbf{1}$ (then projects to the box constraints via $(\mathbf{u} + \mathbf{l})/2$) and applies a single predictor step to land near the central path.
3. **Predictor–Corrector loop**. Uses Mehrotra's predictor–corrector scheme to solve the perturbed KKT system.
4. **Stopping tests**. Monitors primal and dual residuals, together with complementarity.
5. **Infeasibility detection**. Uses a function to measure infeasibility, if it grows too large, the algorithm stops.

4.3 Differences with our implementation.

Our own solver diverges from `quadprog` in two key aspects:

- **Feasible vs. infeasible path.** We enforce strict primal feasibility at each Newton step, whereas `quadprog` allows temporary infeasibility and relies on corrector steps to drive the iterates back toward feasibility.
- **KKT refinement.** `quadprog` performs multiple corrector passes after every predictor step, effectively refining the Newton direction and enabling longer step sizes. Our solver currently uses a single Newton system solve per iteration, which explains the greater iteration count observed in 6.

A further comparison of both approaches —including their practical impact on runtime and solution — is presented in 6.

Chapter 5

Multiclass SVM

5.1 Multiclass Approaches

As we have seen in previous chapters, the Support Vector Machine has been designed for binary classification. However, some strategies have been developed to adapt the model for classification of multiple classes. A comparison can be found in [5].

Experience has shown that constructing several binary classifiers is a better solution than solving a larger optimization problem [2]. The most common approaches are

1. **one-vs-rest (OvR)**, also called one-vs-all: among the classifiers that distinguish between one label and the rest, the one with the highest score assigns the class;

2. **one-vs-one (OvO)**: each binary classification model predicts one class label between every pair of classes and the model with the most predictions or votes is predicted.

While OvO is probably the most common approach in recent implementations [2], we will describe the first approach that was presented in literature, i.e. OvR. Usually the performance are similar. Additionally, we will explore ideas on how solving one training problem in the One-vs-Rest approach may provide useful solutions that may help in solving the others.

5.2 One-vs-Rest approach

As already mentioned, One-vs-Rest is probably the earliest implementation of multiclass SVM. It constructs k SVM models where k is the number of classes, compared for example to OvO where it would be required to build $k(k-1)/2$ classifiers.

The mth SVM is trained using positive labels for the mth class and negative labels for the rest. At the end of all the training sessions, we will obtain k decision functions (2.1).

Then, each sample $\mathbf{x}_i$ will be labeled with the class that resulted in the highest value of its decision function, i.e.

$$\text{class of } \mathbf{x}_i \equiv \text{argmax}_{m=1,\dots,k} f_m(\mathbf{x}_i)$$

In practice, the formulations and solvers described in previous chapter can be used in the loop that traverses the classes to obtain k binary classifiers. Hence, k quadratic problems (2.12) will be solved and then the primal variables will be reconstructed to define the mth classifier.

It is important to mention also some issue related the this approach. In particular, OvR may produce ties (two classifiers vote positive) or rejections (all vote negative).

5.3 Improvements

The ideas described in this section are related to how solving one training problem in the One-vs-Rest approach may provide useful solutions that may help in solving the others. The two main strategies that we propose are the shared kernel matrix and the warm start.

First, it is easy to observe that the apparently independent training sessions of the kth classifiers share the **kernel matrix** (2.5). This fact permits to reduce the training time by pre-computing it before the loop that trains the different models.

Another ideas that is worth mentioning is the **warm start**, i.e. using previous optimal dual variables as initialization for the next classifier's training. If the two classes are similar, we expect similar hyperplanes and therefore a very fast convergence. However, additional care might be required in case of constraints violation.

Chapter 6

Experiments

6.1 Description of Experiments

In this chapter, we deal with input data. We developed the solver by Matlab based on the theoretical parts at previous chapter named Ipsvm. We show our experimental results on the primal-dual interior point method discussing the relation with the underlying theory, and at the end, we will compare the two different solver approaches.
Besides our developed function, we generated also other sub-function to do the ad-hocs tasks like data processing, print results, show plots of results,...The data sets we used are described in detail at 7.3. Before calculation, we normalized all inputs of X values by Z-score normalization. All of the experiments were run on a laptop equipped with CPU 11th Gen Intel Core i5-1145G7 processor running at 2.61 GHz, with four physical cores, and 16Gb of RAM.
We assessed our solver and evaluated their performances with different types of experiments. The Kernel with use for all was Linear Kernel. The starting point was set as the input value of X. The convergence tolerance for Ipsvm (ϵ) was 1e-8, the maximum number of iteration was 200. All data sets were split into training set and test set with the proportion 80:20. The values of C and σ were varied during the experiments.

6.2 Binary class

Firstly, We tested the Ringnorm data. It is an artificial dataset with two equidistributed classes. We did two experiments. The first experiment was to check how fast the method converges during iterations and how long it takes to run. We randomly picked 2000 records from the dataset, keeping the same class proportions as the original data. Then, we used the training set to calculate and solve the problem. We ran tests with two values of C: 1 and 10. The sigma value was changed from 0.3 to 0.7, increasing by 0.05 each time. First, we calculated the optimal points. Then, we used the results to compute the primal model and applied it to predict the labels in order to calculate the accuracy.The purpose is to see how the

speed of convergence and processing time change with different combinations of these starting parameters. The results are shown in the figure 6.1.

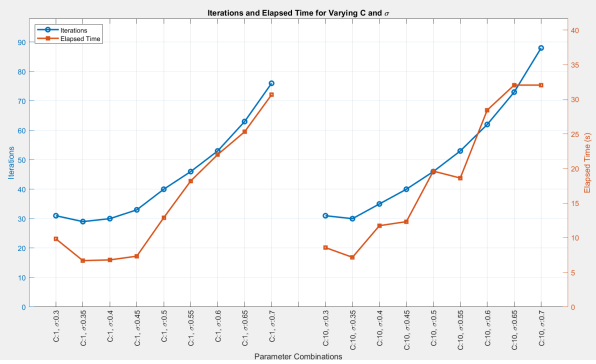

Figure 6.1: Result of convergence with different parameters

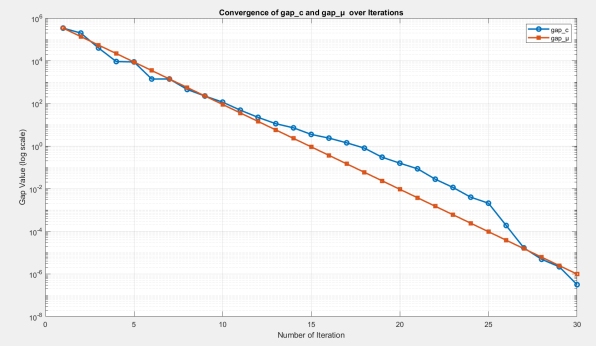

Figure 6.2: Convergence rate at Ringnorm dataset with C=1, σ=0.4

When we implemented the calculation, the model with parameter $C = 1$ gave the accuracy 78% while $C = 10$ gave lower a little bit 77.75%. In both C values, if we increase the σ value, the number of iterations and computation times increase iteratively. It indicates that when σ is higher, the model needs more iterations and takes longer to converge. Furthermore, the computation time at $C = 10$ almost higher than $C = 1$, and the number of iterations was higher too.

The figure 6.2 shows the convergence rate at the $C = 1$ and $\sigma = 0.4$, which are the best parameters with the fast convergence rate and accuracy we have so far.

At the second experiment, we checked how accuracy and convergence rate change with different sizes of datasets. We chose the parameters with $C = 1$ and $\sigma = 0.4$. We ran the size of dataset from 1000 to 7400, with each iteration increase 1000 records. The process to calculate is like the first experiment. The result is showed as table below:

Table 6.1: Performance of the model with different dataset sizes (Ringnorm)

Size	Iterations	Processing Time (s)	Accuracy (%)
1000	29	1.4076	73.50
2000	31	6.4036	76.75
3000	36	40.0516	74.00
4000	38	91.5723	77.50
5000	40	205.7210	76.20
6000	36	244.2114	77.17
7400 (full-size)	37	360.3920	79.12

From Table 6.1, we can see that the processing time increases as the dataset size grows. The increase is nearly linear. When the dataset becomes larger, the processing time becomes much longer. The accuracy is around $74\% - 79\%$ and not

different significantly. In this point, it indicates the model which were generated based on the optimal results give the good performance on this dataset.

Next, we tested the model on two real datasets: German Credit and Pima Indian Diabetes. We ran the calculation 10 times, and the results shown in Table 6.2 are the averages over the 10 experiments. In both datasets, we see that if we increase the C and Sigma (σ) value , the convergence rate and computation time also increase.

Table 6.2: Performance of the model on Two Real Data Sets

Dataset	**Size**	**Features**	**C**	$\boldsymbol{\sigma}$	**Iters**	**Time (s)**
German Credit	1000	20	1	0.4	36	2.502
			1	0.5	37	3.305
			1	0.7	69	4.826
			10	0.4	42	2.672
			10	0.5	43	2.889
			10	0.7	82	6.149
Diabetes	768	8	1	0.4	27	1.040
			1	0.5	37	1.726
			1	0.7	70	2.249
			10	0.4	32	2.486
			10	0.5	42	1.977
			10	0.7	81	3.628

6.3 Small and Medium sized Multi classes

In this experiment, we developed a function to work with multiclass classification. We chose the One-vs-Rest approach, which means we pick one class at a time and train a model to separate that class from the others. Thus, with n classes, we need to build n classification models. The total processing time is the sum of the processing times for each individual class. Finally, the overall accuracy is calculated as the average of the accuracies from each model used for classification. We started with small and medium-sized datasets: the Segment dataset (2310 records, 19 features, 7 classes) and the Iris dataset (150 records, 4 features, 3 classes). We tested several combinations of C and σ values, and the top 3 results are shown in table 6.3. In this experiment, we also used Matlab's quadprog function to compare its performance with our solver. Each test was run 10 times, and we recorded the average results in the table.

Table 6.3: Comparison between Ipsvm and quadprog on Multiclass Datasets

Dataset	C	σ	Ipsvm		quadprog	
			Iteration	Time (s)	Iteration	Time (s)
Segment	1	0.55	426	128.28	128	9.02
	1	0.65	513	143.57	128	7.2
	10	0.60	489	119.37	116	6.7
Iris	1	0.55	137	0.11	24	0.01
	1	0.65	189	0.19	24	0.02
	10	0.60	184	0.16	26	0.04

The models with high values of parameters give the longer processing time. Besides that, the complex data set with higher size and features also take longer time to calculation. Compare the result between Ipsvm function and quadprog. All calculations by Ipsvm took longer time than quadprog. However, for the Segment dataset, Ipsvm gave slightly better results, while for the Iris dataset, one model at $C = 1$, $\sigma = 0.55$ performed better with Ipsvm. Both solvers achieved same accuracy under the same parameter settings. Overall, the results from both methods were quite similar, with only small differences. Figures 6.3, 6.4 show the convergence rate and the 3D plot of the best model on the Iris dataset. In conclusion, the quadprog solver showed a faster convergence rate and shorter processing time on both datasets.

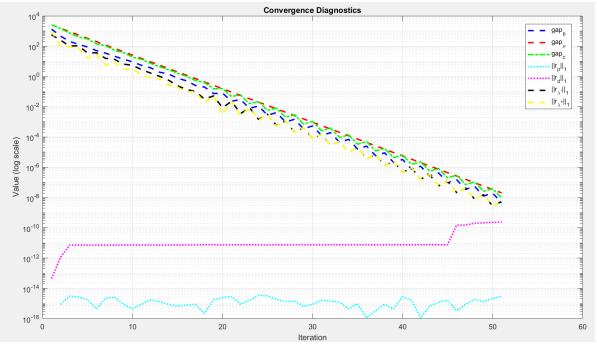

Figure 6.3: Convergence rate at Iris data set

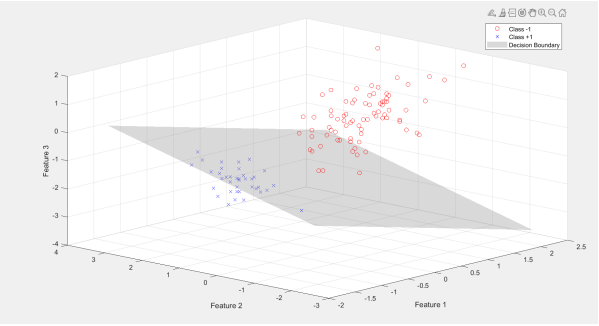

Figure 6.4: 3D Plot of one of the binary classifiers for the Iris Dataset

We calculated the relative gap between the objective function values for each class obtained using the Ipsvm solver and the quadprog solver. The formula we used is:

$$\text{relative gap} = \frac{|F_{\text{ipsvm}} - F_{\text{quadprog}}|}{|F_{\text{quadprog}}|}$$

where F_{ipsvm} are the dual objective function values calculated by Ipsvm solver, F_{quadprog} are the dual objective function values calculated by quadprog solver. However, an similar relative gap can be obtained also using primal objective func-

tions, as it can be verified in table 6.4and in the script `comparing_solvers`.

We selected Segment and Iris, both with parameters $C = 1, \sigma = 0.65$ to evaluate the relative gap for each class. The comparison results are presented in 6.4. In this case, our solver and quadprog solver gave the same value of objective function, the relative gap indicated 0 for all classes in datasets.

Table 6.4: Comparison of Ipsvm and quadprog fvalue results

Dataset	Class	Dual Relative Gap	Primal Relative Gap
Segment, $C = 1$, $\sigma = 0.65$	1	7.18745e-11	2.88093e-11
	2	1.04806e-08	3.39144e-09
	3	1.16724e-11	1.55566e-11
	4	8.50459e-12	1.22756e-08
	5	6.49975e-09	3.13019e-09
	6	4.44314e-10	1.21434e-10
	7	1.06688e-09	2.35080e-09
Iris, $C = 1$, $\sigma = 0.65$	1	1.89636e-10	4.62544e-09
	2	1.78472e-11	5.77988e-10
	3	1.43408e-12	1.59530e-10

6.4 Large multi classes

In this section, we selected the shuttle data set to do the experiment. Because the size is too large (58,000 records, 9 features, 7 classes) so we decided to choose a part of them with 5000 records for the experiment. The sample data was chosen randomly with stratified, make sure the ratio of each class is same as the original part. We did the test with 4 combination of parameters, showed as table 6.5. We also ran the quadprog solver to compare with ours.

Table 6.5: Performance of Ipsvm and quadprog on Shuttle Dataset (5,000 records)

C	σ	Ipsvm			quadprog		
		Iter	Time (s)	Acc. (%)	Iter	Time (s)	Acc. (%)
1	0.6	450	2136.66	94.6	94	113.43	94.6
1	0.7	586	2786.51	94.6	94	99.57	94.6
10	0.65	575	2738.35	94.6	90	106.23	95.2
10	0.7	673	1983.77	94.6	90	57.22	95.2

At Ipsvm solver, lower σ values gave fewer iterations and shorter processing times. For the quadprog solver, both the processing time and number of iterations

were significantly shorter than with Ipsvm. Overall, on the large dataset and across different parameter settings, quadprog produced better results than Ipsvm. The table 6.6 shows the comparison results of 2 solvers, at the paramerters $C = 1, \sigma = 0.6$ and $C = 10, \sigma = 0.7$. It indicates that the objective function values of both solvers show the gap is very small, with the average objective gap at $C = 1, \sigma = 0.6$ is $6.8935e - 07$ and $8.949e - 04$ at $C = 10, \sigma = 0.7$. We conclude that the Ipsvm solver performs well on large multi-class datasets.

Table 6.6: Comparison of Ipsvm and quadprog results at Large dataset

Dataset	**Class**	F_{Ipsvm}	$F_{quadprog}$	**Relative Gap**
	1	-3.74562e+02	-3.74562e+02	4.08646e-11
	2	-6.00000e+00	-6.00000e+00	4.35826e-08
Shuttle,	3	-2.40000e+01	-2.40000e+01	2.60919e-11
$C = 1$,	4	-1.22800e+03	-1.22800e+03	3.04399e-13
$\sigma = 0.6$	5	-1.59724e+00	-1.59724e+00	5.28073e-08
	6	-1.99987e+00	-1.99987e+00	3.38052e-07
	7	-7.96370e-04	-7.96373e-04	4.39068e-06
	1	-3.45358e+03	-3.45358e+03	8.81859e-09
	2	-6.00000e+01	-6.00000e+01	1.21150e-09
Shuttle,	3	-2.40000e+02	-2.40000e+02	1.48124e-12
$C = 10$,	4	-1.22800e+04	-1.22800e+04	1.22367e-12
$\sigma = 0.7$	5	-2.21136e+00	-2.21136e+00	1.62199e-09
	6	-1.99866e+01	-1.99866e+01	2.68183e-07
	7	-7.96368e-04	-7.91410e-04	6.26461e-03

Bibliography

[1] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. en. Cambridge, UK; New York: Cambridge University Press, 2004.

[2] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". en. In: *ACM Transactions on Intelligent Systems and Technology* 2.3 (Apr. 2011), pp. 1–27. ISSN: 2157-6904, 2157-6912. DOI: 10.1145/1961189.1961199.

[3] Jacek Gondzio. "Interior point methods 25 years later". en. In: *European Journal of Operational Research* 218.3 (May 2012), pp. 587–601. ISSN: 03772217. DOI: 10.1016/j.ejor.2011.09.017.

[4] Jacek Gondzio. "Interior Point Methods in Machine Learning". In: *Optimization for Machine Learning*. Ed. by S. Sra, S. Nowozin, and S.J. Wright. MIT Press, 2011, pp. 331–350.

[5] Chih-Wei Hsu and Chih-Jen Lin. "A comparison of methods for multiclass support vector machines". en. In: *IEEE Transactions on Neural Networks* 13.2 (Mar. 2002), pp. 415–425. ISSN: 10459227. DOI: 10.1109/72.991427.

[6] MathWorks. *Quadratic Programming Algorithms MATLAB quadprog*. The MathWorks, Inc. Natick, Massachusetts, 2024. URL: https://it.mathworks.com/help/releases/R2024b/optim/ug/quadratic-programming-algorithms.html.

[7] Kristian Woodsend and Jacek Gondzio. "Exploiting separability in large-scale linear support vector machine training". en. In: *Computational Optimization and Applications* 49.2 (June 2011), pp. 241–269. ISSN: 0926-6003, 1573-2894. DOI: 10.1007/s10589-009-9296-8.

Chapter 7

Appendix

7.1 KKT system of Primal-Dual SVM

The following derivation of the search direction is ours, expanding the lecture slides.

In order to use uppercase letters as diagonal matrices for variables (e.g. Z := diag(z)). Additionally, in this appendix, we follow a common notation in optimization papers, replacing the dual variable α with z, this is necessary to avoid confusion with the matrix A. These changes leads to the following dual optimization problem

$$\begin{array}{ll} \min\limits_{z} & \frac{1}{2} z^T Q z - e^T z \\ \text{subject to} & y^T z = 0 \\ & 0 \leq z \leq Ce \end{array}$$

.

The choices regarding Lagrangian form and multipliers' notation reflect those in [1, §5], i.e. λ_i are the Lagrange multipliers for inequalities, while ν are those related to equalities.

We start from the Lagrangian relaxation

$$\mathcal{L}(z, \lambda_-, \lambda_+, \nu) = \frac{1}{2} z^T Q z - e^T z + \nu^T (y^T z) + \lambda_-^T (-z) + \lambda_+^T (z - Ce)$$

.

Instead of solving the original problem, we aim at iteratively (and approximately) solving a modified problem with a barrier term connected to the inequalities. Given this new "barrier problem" (also called "centering problem"), rather than the original KKT system, the idea is to take one Newton step to solve a perturbed version of the KKT conditions (also called "centrality conditions [1, p. 11.2]") for (7.1), then reduce the perturbation and iteratively solve again until convergence. More in 7.3.

Therefore, the optimality conditions of the centering problem read

$$\begin{aligned}
-z &\leq 0 && \text{primal constraint (i)}\\
z - Ce &\leq 0 && \text{primal constraint (ii)}\\
y^T z &= 0 && \text{primal constraint (iii)}\\
\lambda_- &\geq 0 && \text{dual constraint (i)}\\
\lambda_+ &\geq 0 && \text{dual constraint (ii)}\\
-\lambda_- \cdot (-z) &= \mu e && \text{pert.compl. slackness (i)}\\
-\lambda_+ \cdot (z - Ce) &= \mu e && \text{pert. compl. slackness (ii)}\\
Qz - e - \lambda_- + \lambda_+ + y\nu &= 0 && \text{gradient of Lagrangian} \qquad (7.1)
\end{aligned}$$

Applying the variables' displacement for Newton, we obtain:

$$\begin{aligned}
\Delta z &\geq -z && \text{primal constraint (i)}\\
\Delta z &\leq Ce - z && \text{primal constraint (ii)}\\
y^T \Delta z &= -y^T z && \text{primal constraint (iii)}\\
\Delta\lambda_- &\geq -\lambda_- && \text{dual constraint (i)}\\
\Delta\lambda_+ &\geq -\lambda_+ && \text{dual constraint (ii)}\\
(\lambda_- + \Delta\lambda_-) \circ (z + \Delta z) &= \mu e && \text{pert.compl.slack. (i)}\\
(\lambda_+ + \Delta\lambda_+) \circ (Ce - z - \Delta z) &= \mu e && \text{pert.compl.slack. (ii)}\\
Q\Delta z - \Delta\lambda_- + \Delta\lambda_+ + y\Delta\nu &= && \text{gradient of Lagrangian}\\
= -(Qz - e - \lambda_- + \lambda_+ + y\nu) &
\end{aligned}$$

Inequalities will be considered when taking care of the stepsize. Now, to calculate the step direction, we focus on equalities rearranging with the goal of writing the matrix form of the perturbed KKT, we write

$$\begin{aligned}
Q\Delta z - e - \Delta\lambda_- + \Delta\lambda_+ + y\Delta\nu &= -(Qz - e - \lambda_- + \lambda_+ + y\nu)\\
y^T \Delta z &= -(y^T z)\\
\lambda_- \Delta z + \Delta\lambda_- z &= \mu e - \lambda_- z + \Delta\lambda_- \Delta z\\
-\lambda_+ \Delta z + \Delta\lambda_+ (Ce - z) &= \mu e - \lambda_+ (Ce - z) + \Delta\lambda_+ \Delta z.
\end{aligned}$$

Since we do not aim at an exact solution of the system, for computing a Newton step, we will use a linearized version of the KKT removing the bilinear terms $\Delta\lambda_-\Delta z$ and $\Delta\lambda_+\Delta z$. In the following system and in later steps, note that $U := diag(Ce)$ and $A = y^T$:

$$\begin{bmatrix} Q & A^T & -I & I \\ A & 0 & 0 & 0 \\ \Lambda_- & 0 & Z & 0 \\ -\Lambda_+ & 0 & 0 & U - Z \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\nu \\ \Delta\lambda_- \\ \Delta\lambda_+ \end{bmatrix} = \begin{bmatrix} r_z \\ r_{nu} \\ r_{lm} \\ r_{lp} \end{bmatrix}$$

Given that we removed the bilinear terms, we have:

$$\begin{aligned} r_z &= -(Qz - e - \lambda_- + \lambda_+ + y^T\nu) \\ r_\nu &= -(y^T z) \\ r_{lm} &= -(\lambda_- z - \mu e) \\ r_{lp} &= -(\lambda_+(U - z) - \mu e). \end{aligned}$$

We can start solving from the last two constraints over $\Delta\lambda_-$ and $\Delta\lambda_+$, yielding

$$\begin{aligned} \Delta\lambda_- &= \mu Z^{-1}e - Z^{-1}\Lambda_-\Delta z - \lambda_- \\ \Delta\lambda_+ &= \mu(U - Z)^{-1}e + (U - Z)^{-1}\Lambda_+\Delta z - \lambda_+. \end{aligned}$$

Substituting into the first line of the matrix 7.1 and, given $r_z = -(Qz - e - \lambda_- + \lambda_+ + y^T\nu)$, we will simplify λ_- and λ_+ from both sides

$$\begin{aligned} & Q\Delta z + y^T\Delta\nu \\ & \quad - \Delta\lambda_- + \Delta\lambda_+ && = r_z \\ & Q\Delta z + y^T\Delta\nu \\ & \quad - \mu Z^{-1}e + Z^{-1}\Lambda_-\Delta z + \lambda_- \\ & \quad + \mu(U - Z)^{-1}e + (U - Z)^{-1}\Lambda_+\Delta z - \lambda_+ && = r_z \\ & Q\Delta z + y^T\Delta\nu \\ & \quad + Z^{-1}\Lambda_-\Delta z + (U - Z)^{-1}\Lambda_+\Delta z && = r_c, \end{aligned}$$

where

$$r_c = -\left(Qz - e + y^T\nu - \mu Z^{-1}e + \mu(U - Z)^{-1}e\right)$$

.

Together with the second line of the matrix 7.1, we build the augmented system equations as

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} r_c \\ r_\nu \end{bmatrix}$$

Where

$$\begin{aligned} H &= Q + Z^{-1}\Lambda_- + (U - Z)^{-1}\Lambda_+ \\ A &= y^T \\ r_c &= -(Qz - e + y^T\nu - \mu Z^{-1}e + \mu(U - Z)^{-1}e) \\ r_\nu &= -(y^T z) \end{aligned}$$

We observe that the term $H - Q$ is diagonal and strictly positive, hence H is strictly positive definite and nonsingular.

Also, we can work on r_c using some considerations. In the central path, we expect the stationarity constraint to be respected (7.1) (i.e. $\nabla\mathcal{L}(z, \lambda_-, \lambda_+, \nu) = 0$, meaning $Qz - e + y^T\nu = +\lambda_- - \lambda_+$. Hence, we can rewrite r_c as

$$\begin{aligned} r_c &= -(Qz - e + y^T\nu - \mu Z^{-1}e + \mu(U - Z)^{-1}e) \\ &= -(\lambda_- - \lambda_+ - \mu Z^{-1}e + \mu(U - Z)^{-1}e) \\ &= -\lambda_- + \lambda_+ + \mu Z^{-1}e - \mu(U - Z)^{-1}e \\ &= -\lambda_- + \lambda_+ + \gamma \end{aligned}$$

Furthermore, the precomputed $\gamma = \mu Z^{-1}e - \mu(U - Z)^{-1}e$ just introduced in the steps above, we be written in a different form that can greatly improve numerical stability,

$$\begin{aligned} \gamma_i &= \mu \cdot (\frac{1}{z_i} - \frac{1}{u_i - z_i}) \\ &= \mu \cdot \frac{u_i - z_i - z_i}{z \cdot (u_i - z_i)} \\ &= \mu \cdot \frac{u_i - 2z_i}{z \cdot (u_i - z_i)} \end{aligned} \tag{7.2}$$

.

The augmented system can be further simplified obtaining the normal equation $M\Delta\nu = r_b$ eliminating Δz as follows

$$\begin{aligned} H\Delta z + A^T\Delta\nu &= r_c \\ A\Delta z &= AH^{-1}(r_c - A^T\Delta\nu) \\ r_\nu &= AH^{-1}r_c - AH^{-1}A^T\Delta\nu \\ AH^{-1}A^T\Delta\nu &= AH^{-1}r_c - r_\nu \\ M\Delta\nu &= r_b \end{aligned}$$

Using the block elimination, where emerged the Schur's Complement $M := AH^{-1}A^T$, we obtained $M\Delta\nu = r_b$.

In some papers, such as [7], the matrix H is written as $Q + \Theta^{-1}$, given $\Theta^{-1} := Z^{-1}\Lambda_- + (U - Z)^{-1}\Lambda_+$, which leads to write $M := A(Q + \Theta^{-1})^{-1}A^T$.

Considering the stable form of γ (7.2), we can write the terms of the normal equation $M\Delta\nu = r_b$ as

$$\begin{aligned} M &= AH^{-1}A^T \\ r_b &= AH^{-1}r_c + A^Tz \end{aligned}$$

Once $\Delta\nu$ is computed, the other deltas can be obtained looking back at (7.1) and (7.1)

$$\begin{aligned} \Delta z &= H^{-1}(r_c - A^T\Delta\nu) \\ \Delta\lambda_- &= \mu Z^{-1}e - Z^{-1}\Lambda_-\Delta z - \lambda_- \\ \Delta\lambda_+ &= \mu(U - Z)^{-1}e + (U - Z)^{-1}\Lambda_+\Delta z - \lambda_+ \end{aligned}$$

7.2 Centering problem

In this section of the appendix, we explore the differences in the gaps that are mentioned, i.e. the true duality gap, the surrogate gaps (centrality gap)and the μ-gap).
In particular, the idea of central path is explored to get a deeper understanding of how the primal-dual method is derived from the barrier method.

Guided by [1, §11.2.2], we derive the surrogate gaps that are used as convergence conditions. Consider the centering problem, obtained from (3.1)

$$\begin{array}{ll} \min\limits_{x} & \frac{1}{\mu} f_0(x) + \phi(x) \\ \text{subject to} & Ax = b, \end{array} \tag{7.3}$$

where we define the barrier function

$$\phi(x) = -\sum_{i=1}^{m} log(-f_i(x)),$$

its derivative

$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{-f_i(x)} \nabla f_i(x)$$

and μ as the "barrier hyperparameter", that in the primal-dual is more appropriately called "centering parameter".

This new equality constrained optimization problem has the following necessary and sufficient optimality conditions, with the optimal solution $x^*(\mu)$

$$\begin{array}{ll} i) & Ax^*(\mu) = b \\ ii) & \exists\hat{\nu} \in \mathbb{R}^p \text{ s.t.} \\ & 0 = f_0(x^*(\mu)) + \mu \sum_{i=1}^{m} \frac{1}{-f_i(x^*(\mu))} \nabla f_i(x^*(\mu) + \mu A^T \hat{\nu}\,. \end{array}$$

We define the pair of dual variables

$$\begin{aligned} \lambda_i^*(\mu) &= -\mu \frac{1}{f_i(x^*(\mu))} \\ \nu_i^*(\mu) &= \mu\hat{\nu} \end{aligned}$$

and it can be shown that they are dual feasible by writing the previous optimality conditions with these variables. This step is crucial because if every central point yields a dual feasible point, then it can be used as a lower bound on the optimal value p^*.

Given $x^*(\mu)$ solution of the centering problem, dual function can be expressed

as

$$\begin{aligned} g(\lambda^*(\mu), \nu^*(\mu)) &= f_0(x^*(\mu)) + \sum_{i=1}^{m} \lambda_i^*(\mu) f_i(x^*(\mu)) + \nu^*(\mu)(Ax^*(\mu) - b) \\ &= f_0(x^*(\mu)) + \sum_{i=1}^{m} \left(-\mu \frac{1}{f_i(x^*(\mu))} \right) f_i(x^*(\mu)) + \mu\hat{\nu}(Ax^*(\mu) - b) \\ &= f_0(x^*(\mu)) + \sum_{i=1}^{m} -\mu + \mu\hat{\nu} \cdot 0 \\ &= f_0(x^*(\mu)) - m \cdot \mu \,. \end{aligned}$$

The last equation shows that, if $x^*(\mu)$ is the exact solution of the centering problem (7.3), the gap can be obtained simply as

$$gap_\mu = m \cdot \mu \,.$$

Since we do not aim at an exact centering step, the safer way to compute this surrogate gap (or centering gap) through the iterations is clearly

$$gap_c = \sum_{i=1}^{m} \lambda_i^*(\mu) f_i(x^*(\mu)) + \nu^*(\mu)(Ax^*(\mu) - b) \,.$$

However, gap_μ can still be seen as a target and used to tune the hyperparameter μ.

In order to understand how well the centering step is computed, and hence how close we are to the central path, we might look at residuals of (3.4). In particular, when the residuals are close to zero, we expect $gap_c \approx gap_\mu$. In practice, they cannot be the same, since the KKT system is linearized and an exact solution for the problem cannot be achieved.

How is the gap of the centering problem (7.3) related to the gap of original problem (3.1)? Weak duality states that the dual function $g(\lambda, \nu)$ set a lower bound for $f_0(x)$, then for every feasible $\lambda \geq 0$

$$\begin{aligned} g(\lambda, \nu) &\leq f_0(x) \\ g(\lambda^*(\mu), \nu^*(\mu)) &\leq f_0(x^*) \\ f_0(x^*(\mu)) + \text{gap}_c &\leq f_0(x^*) \\ f_0(x^*(\mu)) - f_0(x^*) &\leq \text{gap}_c \,. \end{aligned}$$

We just showed that $gap_c = \sum_{i=1}^{m} \lambda_i^*(\mu) f_i(x^*(\mu)) + \nu^*(\mu)(Ax^*(\mu) - b)$ can be used as an upper bound for the duality gap of the original problem. Once the centering problem is solved well enough, gap_c could be replaced by gap_μ.

From a KKT system perspective, the only difference between the KKT conditions of the original problem and the centrality conditions (3.4) is that the complementarity condition $-\lambda_i f_i(x) = 0$ is replaced by the condition $-\lambda_i f_i(x) = \mu$. In particular, for small μ, $x^*(\mu)$ and the associated dual point $\lambda^*(\mu), \nu^*(\mu)$ 'almost'

satisfy the KKT optimality conditions of the original problem.

To sum up, in interior-point methods, several notions of optimality gap are used. The gap_μ, defined as $m \cdot \mu$, holds when $x^*(\mu)$ exactly solves the centering problem; it represents the ideal duality gap along the central path. In practice, however, iterates are not exactly central. The *centrality gap*, given by $\sum_{i=1}^{m} \lambda_i f_i(x) + \nu^T(Ax - b)$, evaluates the residual gap at the current iterate and is used as a stopping criterion. Finally, the true duality gapgap_p is defined as $f_0(x) - g(\lambda, \nu)$ when the dual function is evaluated at a feasible (λ, ν). Only when (x, λ, ν) lie on the central path do all three expressions coincide.

7.3 Data sets

To investigate what performance results can be expected in application. We used 4 standard data sets.

- **Ringnorm**: As introduced by Breiman (1998), this dataset contains 7,400 instances described by 20 numerical features. The data are divided into two equidistributed classes, each generated from a multivariate normal distribution with different variance-covariance matrices.
- **Pima Indian Diabetes**: This real-world medical dataset contains 768 records with 7 features and binary class labels. It was from the UCI Machine Learning Repository and is commonly used for evaluating classification models on realistic data.
- **German Credit**: a dataset contains on 20 variables and the classification whether an applicant is considered a Good or a Bad credit risk for 1000 loan applicants. It is a dataset used for SVM evaluation and has been preprocessed and distributed via the LibSVM dataset collection.
- **Segment**: a dataset is an image segmentation database similar to a database already present in the repository (Image segmentation database) but in a slightly different form. It has 2310 records, with 19 features and 7 classes.
- **Iris**: Derived from the classic Iris dataset. This subset includes 150 records with 4 features, restricted to three classes for binary classification tasks.
- **Shuttle**: a dataset has 58,000 records, with 9 features and 7 classes. Mainly in type of numerical data. It was propsed by StatLog and all data were processed by LibSVM dataset collection.

All datasets are publicly available from the LibSVM data repository: `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.