OTDM - Constrained Optimization - SVM

Julian Fransen, Danila Kokin

2024-11-24

Contents

Section 1: implement SVM in AMPL

In this first section, we generate data using the generators given, and use this as a use case for SVMs. We implement both the primal and dual in AMPL.

Data generation and preprocessing

We generate the data using `gensvdat`, where we use our 2 students identifier numbers for the seed for training and test, e.g. :

```
./gensvmdat tes_raw.dat 100 4624        # te =  test, s = small -> tes
./gensvmdat trl_raw.dat 100 7438042     # tr = train, l = large -> trl
```

We create 4 files in total: a small set of 100 points and a large one of 100k points. We use the same size for testing and training. After creating the raw data files, we utilize a shell script which performs processing so make sure that the data files can be loaded into AMPL. This consist of adding a header including variables `m` and `n`, removing the `*` symbol and displaying in terminal the number of misclassifications. Finally, we copy the processed data files to the `primal/` and `dual/` directories.

Primal SVM Problem

We aim to solve the following optimization problem:

$$\min_{w,\gamma,s} \frac{1}{2} w^T w + \nu e^T s$$

subject to:

$$Y(Aw + \gamma e) + s \geq e,$$
$$s \geq 0,$$

where:

- **Decision Variables**:
$$(w, \gamma, s) \in \mathbb{R}^{n+1+m},$$
  - $w \in \mathbb{R}^n$: Weight vector for the hyperplane.
  - $\gamma \in \mathbb{R}$: Bias term.
  - $s \in \mathbb{R}^m$: Slack variables for handling misclassifications.
- **Constants**:
  - $\nu > 0$: Regularization parameter controlling the trade-off between margin size and misclassification penalty.
  - $A \in \mathbb{R}^{m \times n}$: Matrix where rows represent feature vectors of the data points.
  - $Y \in \mathbb{R}^{m \times m}$: Diagonal matrix of labels, where $Y_{ii} = y_i$, and $y_i \in \{-1, 1\}$.
  - $e \in \mathbb{R}^m$: Vector of ones ($e = [1, 1, \ldots, 1]^T$).

Dual SVM Formulation

We formulate the dual SVM model using explicit indices instead of matrix notation.

Objective Function

$$\max_{\lambda} \quad \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j \left(\sum_{k=1}^{n} A_{ik} A_{jk} \right)$$

subject to: 1.

$$\sum_{i=1}^{m} \lambda_i y_i = 0$$

2.

$$0 \leq \lambda_i \leq \nu, \quad \forall i = 1, \ldots, m$$

where:

- λ_i: Dual variable for the i-th data point.
- y_i: Label of the i-th data point (± 1).
- A_{ik}: Feature k of the i-th data point.
- ν: Regularization parameter.

The Separation Hyperplane in SVM

The separation hyperplane in a Support Vector Machine (SVM) is determined by $\mathbf{w}$ (the weight vector) and γ (the bias term). It defines the decision boundary that separates the two classes. The equation of the hyperplane is:

$$\sum_{j=1}^{n} w_j x_j + \gamma = 0$$

where: - $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ is the weight vector, - γ is the bias term, - $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ represents the coordinates of a point in n-dimensional space.

The decision rule for classification is:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{n} w_j x_j + \gamma\right)$$

- If $f(\mathbf{x}) > 0$, classify as $+1$.
- If $f(\mathbf{x}) < 0$, classify as -1.

Since the separation hyperplane only depends on `w` and γ, we know that the if these are the same for the primal and dual, then the hyperplane will be the same as well. To find the the values of `w` and γ for the dual problem, we use these formulas:

$$w = \sum_{i=1}^{m} \lambda_i y_i \phi(x_i)$$

where ϕ is the identity matrix, in this particular case.

$$\gamma = y_k - \sum_{j=1}^{n} w_j \cdot A_{k,j}$$

where `k` is the index of the first support vector. We will find support vectors based on the property: $0 < \lambda_k < \nu$ for all $k \in SV$.

Train:

Primal

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/primal
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
model primal.mod;
data trs.dat;
let nu := 0.9;
solve;
display n, gamma, w;
display n, gamma, w > sparams.dat;
quit;
EOF
```

Small dataset

```
## CPLEX 22.1.1.0: optimal solution; objective 45.1830374
## 11 separable QP barrier iterations
## No basis.
## n = 4
## gamma = -3.65763
##
## w [*] :=
## 1  1.28474
## 2  1.97146
## 3  2.35625
## 4  2.05919
## ;
```

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/primal
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
model primal.mod;
data trl.dat;
let nu := 0.9;
solve;
display n, gamma, w;
display n, gamma, w > lparams.dat;
quit;
EOF
```

Large dataset

```
## CPLEX 22.1.1.0: optimal solution; objective 26542.77347
## 14 separable QP barrier iterations
## No basis.
## n = 4
## gamma = -10.1811
##
## w [*] :=
## 1  5.0655
## 2  5.09634
## 3  5.12616
## 4  5.08233
## ;
```

Dual

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/dual
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
model dual.mod;
data trs.dat;
let nu := 0.9;
```

```
solve;
param w {j in 1..n};
for {j in 1..n} {
    let w[j] := sum {i in 1..m} lambda[i] * y[i] * A[i, j];
}
param svi := 2;
param gamma := y[svi] - sum {j in 1..n} w[j] * A[svi, j];
display w, gamma;
display n, gamma, w > sparams.dat;
quit;
EOF
```

Small dataset

```
## CPLEX 22.1.1.0: optimal solution; objective 45.18303736
## 12 QP barrier iterations
## No basis.
## w [*] :=
## 1  1.28474
## 2  1.97146
## 3  2.35625
## 4  2.05919
## ;
##
## gamma = -3.65763
```

Large dataset

{bash} cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/dual /Users/danilakokin/Download <<EOF option solver cplex; model dual.mod; data trl.dat; let nu := 0.9; solve; param w {j in 1..n}; for {j in 1..n} { let w[j] := sum {i in 1..m} lambda[i] * y[i] * A[i, j]; } param svi := 2; param gamma := y[svi] - sum {j in 1..n} w[j] * A[svi, j]; display w, gamma; display n, gamma, w > lparams.dat; quit; EOF ####

{bash} cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/dual chmod +x fix_params.sh ./fix_params.sh lparams.dat #### As you can see, the values for the objective function, `w*` and γ are identical (at least up to 5 decimals) for the dual and the primal, which means they found both exactly the same optimal hyperplane. This is consistent with theory: the dual should be exactly the same as the primal, except with fewer constraints.

Evaluation

Primal

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/primal
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
```

```
model eval.mod;
data tes.dat;
data sparamsformated.dat;
display accuracy, precision, recall, f1_score;
quit;
EOF
```

Evaluation on small dataset

```
## accuracy = 0.88
## precision = 0.842105
## recall = 0.941176
## f1_score = 0.888889
```

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/primal
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
model eval.mod;
data tel.dat;
data lparamsformated.dat;
display accuracy, precision, recall, f1_score;
quit;
EOF
```

Evaluation on large dataset

```
## accuracy = 0.94773
## precision = 0.946665
## recall = 0.948955
## f1_score = 0.947809
```

Dual

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/dual
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
model eval.mod;
data tes.dat;
data sparamsformated.dat;
display accuracy, precision, recall, f1_score;
quit;
EOF
```

Evaluation on small dataset

```
## accuracy = 0.88
## precision = 0.842105
## recall = 0.941176
## f1_score = 0.888889
```

Evaluation on large dataset

```
{bash} cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/dual /Users/danilakokin/Download
<<EOF option solver cplex; model eval.mod; data tel.dat; data lparamsformated.dat; display
accuracy, precision, recall, f1_score; quit; EOF ####
```

Section 2: applying SVMs to new dataset.

```r
# Step 1: Load the data
df <- read.csv("/Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/data_formatter/money.csv")

# Step 2: Convert 'class' column to binary variable (1 and -1)
df$class <- ifelse(df$class == 1, 1, -1)

# Step 3: Remove the 'id' column
df$id <- NULL

# Step 4: Split the data into train and test sets
set.seed(123)  # For reproducibility
trainll <- sample(1:nrow(df), size = nrow(df) / 2)  # Randomly sample 50% of the rows for training
testll <- setdiff(1:nrow(df), trainll)  # The remaining rows for testing

train <- df[trainll, ]
test <- df[testll, ]

# Step 5: Normalize train data (except target)
# Identify numeric columns (excluding 'class')
numeric_cols <- names(train)[sapply(train, is.numeric) & names(train) != "class"]

# Compute min and max values for each numeric column in the training set
min_vals <- sapply(train[, numeric_cols], min, na.rm = TRUE)
max_vals <- sapply(train[, numeric_cols], max, na.rm = TRUE)

# Define normalization function
normalize <- function(x, min_val, max_val) {
  (x - min_val) / (max_val - min_val)
}

# Apply normalization to the train set
for (col_name in numeric_cols) {
  min_val <- min_vals[col_name]
  max_val <- max_vals[col_name]
  train[[col_name]] <- normalize(train[[col_name]], min_val, max_val)
}

# Step 6: Normalize test data (except target) based on the train normalizer
for (col_name in numeric_cols) {
  min_val <- min_vals[col_name]
  max_val <- max_vals[col_name]
  test[[col_name]] <- normalize(test[[col_name]], min_val, max_val)
}
```

```r
# Step 7: Display summary and row counts for train and test sets
summary(train)
```

```
##     variance         kurtosis         skewness         entropy      
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.3786   1st Qu.:0.4594   1st Qu.:0.1532   1st Qu.:0.5596  
##  Median :0.5488   Median :0.6009   Median :0.2503   Median :0.7261  
##  Mean   :0.5420   Mean   :0.5878   Mean   :0.2865   Mean   :0.6656  
##  3rd Qu.:0.7066   3rd Qu.:0.7695   3rd Qu.:0.3612   3rd Qu.:0.8229  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
##      class        
##  Min.   :-1.0000  
##  1st Qu.:-1.0000  
##  Median :-1.0000  
##  Mean   :-0.1312  
##  3rd Qu.: 1.0000  
##  Max.   : 1.0000
```

```r
cat("Number of rows in train set:", nrow(train), "\n\n")
```

```
## Number of rows in train set: 686
```

```r
summary(test)
```

```
##     variance             kurtosis           skewness            entropy        
##  Min.   :-0.0004112   Min.   :0.004258   Min.   :-0.003033   Min.   :-0.07669  
##  1st Qu.: 0.3809555   1st Qu.:0.446418   1st Qu.: 0.163883   1st Qu.: 0.52343  
##  Median : 0.5317613   Median :0.602478   Median : 0.252592   Median : 0.72085  
##  Mean   : 0.5358386   Mean   :0.586773   Mean   : 0.285057   Mean   : 0.66009  
##  3rd Qu.: 0.7150139   3rd Qu.:0.771197   3rd Qu.: 0.367578   3rd Qu.: 0.82126  
##  Max.   : 0.9811344   Max.   :0.991715   Max.   : 0.986078   Max.   : 1.02885  
##      class         
##  Min.   :-1.00000  
##  1st Qu.:-1.00000  
##  Median :-1.00000  
##  Mean   :-0.09038  
##  3rd Qu.: 1.00000  
##  Max.   : 1.00000
```

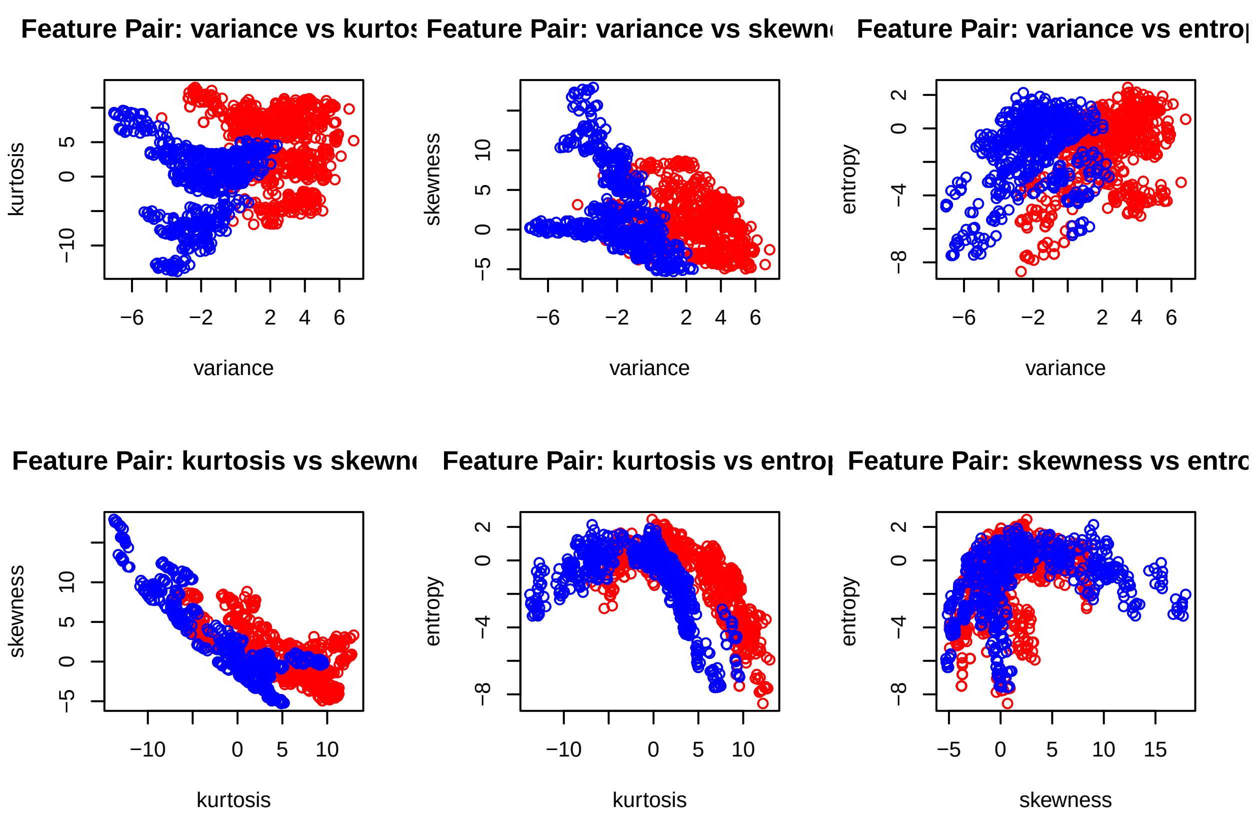
```r
cat("Number of rows in test set:", nrow(test), "\n")
```

```
## Number of rows in test set: 686
```

```r
train_file <- "/Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/data_formatter/trm.csv"
test_file <- "/Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/data_formatter/tem.csv"

write.csv(train, file = train_file, row.names = FALSE)
write.csv(test, file = test_file, row.names = FALSE)
```

```
df[,1:4]
```

```
data <- df
features <- colnames(data)[1:4]
target <- colnames(data)[5]
num_features <- length(features)

par(mfrow = c(2, 3))  # Set up a plotting grid

for (i in 1:(num_features - 1)) {
  for (j in (i + 1):num_features) {
    # Plot each feature pair
    plot(data[[features[i]]], data[[features[j]]],
         col = ifelse(data[,target] == 1, "blue", "red"),
         xlab = features[i], ylab = features[j],
         main = paste("Feature Pair:", features[i], "vs", features[j]))
  }
}
```



Primal

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/primal
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
```

```
option solver cplex;
model primal.mod;
data trm.dat;
let nu := 0.9;
solve;
display n, gamma, w;
display n, gamma, w > mparams.dat;
quit;
EOF
```

```
## CPLEX 22.1.1.0: optimal solution; objective 106.7440306
## 19 separable QP barrier iterations
## No basis.
## n = 4
## gamma = 7.59591
##
## w [*] :=
## 1  -5.98142
## 2  -5.26369
## 3  -5.58756
## 4   0.132916
## ;
```

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/primal
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
model eval.mod;
data tem.dat;
data mparamsformated.dat;
display true_positive, true_negative, false_positive, false_negative;
display accuracy, precision, recall, f1_score;
quit;
EOF
```

```
## true_positive = 312
## true_negative = 360
## false_positive = 14
## false_negative = 0
##
## accuracy = 0.979592
## precision = 0.957055
## recall = 1
## f1_score = 0.978056
```

Dual

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/dual
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
model dual.mod;
data trm.dat;
```

```
let nu := 0.9;
solve;
param w {j in 1..n};
for {j in 1..n} {
    let w[j] := sum {i in 1..m} lambda[i] * y[i] * A[i, j];
}
param svi := 2;
param gamma := y[svi] - sum {j in 1..n} w[j] * A[svi, j];
display gamma, w;
display n, gamma, w > mparams.dat;
quit;
EOF
```

```
## CPLEX 22.1.1.0: optimal solution; objective 106.7440305
## 21 QP barrier iterations
## No basis.
## gamma = 8.978
##
## w [*] :=
## 1  -5.98142
## 2  -5.26369
## 3  -5.58756
## 4   0.132916
## ;
```

```
cd /Users/danilakokin/Desktop/UPC/Semester3/OTDM/OTDM_Project_2/dual
/Users/danilakokin/Downloads/ampl_macos64/ampl <<EOF
option solver cplex;
model eval.mod;
data tem.dat;
data mparamsformated.dat;
display true_positive, true_negative, false_positive, false_negative;
display accuracy, precision, recall, f1_score;
quit;
EOF
```

```
## true_positive = 312
## true_negative = 185
## false_positive = 189
## false_negative = 0
##
## accuracy = 0.72449
## precision = 0.622754
## recall = 1
## f1_score = 0.767528
```