Matrix methods – Support vector machines

MATH 2740 – Mathematics of Data Science – Lecture 13

Julien Arino

julien.arino@umanitoba.ca

Department of Mathematics @ University of Manitoba

Fall 202X

The University of Manitoba campuses are located on original lands of Anishinaabeg, Ininew, Anisininew, Dakota and Dene peoples, and on the National Homeland of the Red River Métis. We respect the Treaties that were made on these territories, we acknowledge the harms and mistakes of the past, and we dedicate ourselves to move forward in partnership with Indigenous communities in a spirit of Reconciliation and collaboration.

Outline

Support vector machines (SVM)

Support vector machines (SVM)

We are given a training dataset of n points of the form

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$$

where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i = \{-1, 1\}$. The value of y_i indicates the class to which the point $\boldsymbol{x}_i$ belongs

We want to find a **surface** $\mathcal{S}$ in $\mathbb{R}^p$ that divides the group of points into two subgroups

Once we have this surface $\mathcal{S}$, any additional point that is added to the set can then be *classified* as belonging to either one of the sets depending on where it is with respect to the surface $\mathcal{S}$

Linear SVM

We are given a training dataset of n points of the form

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$$

where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i = \{-1, 1\}$. The value of y_i indicates the class to which the point $\boldsymbol{x}_i$ belongs

> ***Linear SVM*** *– Find the "maximum-margin hyperplane" that divides the group of points $\boldsymbol{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is such that the distance between the hyperplane and the nearest point $\boldsymbol{x}_i$ from either group is maximized.*

Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are the **support vectors**

Any **hyperplane** can be written as the set of points **x** satisfying

$$\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} - b = 0$$

where $\boldsymbol{w}$ is the (not necessarily normalized) **normal vector** to the hyperplane (if the hyperplane has equation $a_1 z_1 + \cdots + a_p z_p = c$, then $(a_1, \ldots, a_n)$ is normal to the hyperplane)

The parameter $b/\|\boldsymbol{w}\|$ determines the offset of the hyperplane from the origin along the normal vector $\boldsymbol{w}$

Remark: a hyperplane defined thusly is not a subspace of $\mathbb{R}^p$ unless $b = 0$. We can of course transform the data so that it is...

Linearly separable points

Let X_1 and X_2 be two sets of points in $\mathbb{R}^p$

Then X_1 and X_2 are **linearly separable** if there exist $w_1, w_2, .., w_p, k \in \mathbb{R}$ such that

- every point $x \in X_1$ satisfies $\sum_{i=1}^{p} w_i x_i > k$
- every point $x \in X_2$ satisfies $\sum_{i=1}^{p} w_i x_i < k$

where x_i is the ith component of x

Hard-margin SVM

If the training data is **linearly separable**, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible

The region bounded by these two hyperplanes is called the “margin”, and the maximum-margin hyperplane is the hyperplane that lies halfway between them

With a normalized or standardized dataset, these hyperplanes can be described by the equations

- $\mathbf{w}^\mathsf{T}\mathbf{x} - b = 1$ (anything on or above this boundary is of one class, with label 1)
- $\mathbf{w}^\mathsf{T}\mathbf{x} - b = -1$ (anything on or below this boundary is of the other class, with label -1)

Distance between these two hyperplanes is $2/\|\boldsymbol{w}\|$

$\Rightarrow$ to maximize the distance between the planes we want to minimize $\|\boldsymbol{w}\|$

The distance is computed using the distance from a point to a plane equation

We must also prevent data points from falling into the margin, so we add the following constraint: for each i either

$$\mathbf{w}^{\mathsf{T}}\mathbf{x}_i - b \geq 1\,,\ \text{if } y_i = 1$$

or

$$\mathbf{w}^{\mathsf{T}}\mathbf{x}_i - b \leq -1\,,\ \text{if } y_i = -1$$

(Each data point must lie on the correct side of the margin)

This can be rewritten as

$$y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n$$

or

$$y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b) - 1 \geq 0, \quad \text{for all } 1 \leq i \leq n$$

We get the optimization problem:

Minimize $\|\mathbf{w}\|$ *subject to* $y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b) - 1 \geq 0$ *for* $i = 1, \ldots, n$

The **w** and b that solve this problem determine the classifier, $\mathbf{x} \mapsto \mathrm{sgn}(\mathbf{w}^\mathsf{T}\mathbf{x} - b)$ where $\mathrm{sgn}(\cdot)$ is the **sign function.**

The maximum-margin hyperplane is completely determined by those $\boldsymbol{x}_i$ that lie nearest to it

These $\boldsymbol{x}_i$ are the **support vectors**

Writing the goal in terms of Lagrange multipliers

Recall that our goal is to

$$\textit{minimize } \|\mathbf{w}\| \textit{ subject to } y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b) - 1 \geq 0 \textit{ for } i = 1, \ldots, n$$

Using Lagrange multipliers $\lambda_1, \ldots, \lambda_n$, we have the function

$$L_P := F(\boldsymbol{w}, b\lambda_1, \ldots, \lambda_n) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \lambda_i y_i(\boldsymbol{x}_i\boldsymbol{w} + b) + \sum_{i=1}^{n} \lambda_i$$

Note that we have as many Lagrange multipliers as there are data points. Indeed, there are that many inequalities that must be satisfied

The aim is to minimise L_p with respect to $\boldsymbol{w}$ and b while the derivatives of L_p w.r.t. λ_i vanish and the $\lambda_i \geq 0$, $i = 1, \ldots, n$

Lagrange multipliers

We have already seen Lagrange multipliers, when we were studying PCA

Maximisation using Lagrange multipliers (V1.0)

We want the max of $f(x_1, \ldots, x_n)$ under the constraint $g(x_1, \ldots, x_n) = k$

1. Solve

$$\begin{aligned} \nabla f(x_1, \ldots, x_n) &= \lambda \nabla g(x_1, \ldots, x_n) \\ g(x_1, \ldots, x_n) &= k \end{aligned}$$

   where $\nabla = (\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n})$ is the **gradient operator**
2. Plug all solutions into $f(x_1, \ldots, x_n)$ and find maximum values (provided values exist and $\nabla g \neq \mathbf{0}$ there)

λ is the **Lagrange multiplier**

The gradient

$f : \mathbb{R}^n \to \mathbb{R}$ function of several variables, $\nabla = \left(\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_n}\right)$ the gradient operator

Then

$$\nabla f = \left(\frac{\partial}{\partial x_1} f, \dots, \frac{\partial}{\partial x_n} f\right)$$

So ∇f is a *vector-valued* function, $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$; also written as

$$\nabla f = f_{x_1}(x_1, \dots, x_n)\boldsymbol{e}_1 + \cdots f_{x_n}(x_1, \dots, x_n)\boldsymbol{e}_n$$

where f_{x_i} is the partial derivative of f with respect to x_i and $\{\boldsymbol{e}_1, \dots, \boldsymbol{e}_n\}$ is the standard basis of $\mathbb{R}^n$

Lagrange multipliers (V2.0)

However, the problem we were considering then involved a single multiplier λ

Here we want $\lambda_1, \ldots, \lambda_n$

Lagrange multiplier theorem

Theorem 1

Let $f : \mathbb{R}^n \to \mathbb{R}$ be the objective function, $g : \mathbb{R}^n \to \mathbb{R}^c$ be the constraints function, both being C^1. Consider the optimisation problem

$$\begin{aligned} \textit{maximize } & f(x) \\ \textit{subject to } & g(x) = 0 \end{aligned}$$

Let x^* be an optimal solution to the optimization problem, such that $\operatorname{rank}(Dg(x^*)) = c < n$, where $Dg(x^*)$ denotes the matrix of partial derivatives

$$\left[\partial g_j / \partial x_k\right]$$

Then there exists a unique Lagrange multiplier $\lambda^* \in \mathbb{R}^c$ such that

$$Df(x^*) = {\lambda^*}^T Dg(x^*)$$

Lagrange multipliers (V3.0)

Here we want $\lambda_1, \ldots, \lambda_n$

But we also are looking for $\lambda_i \geq 0$

So we need to consider the so-called Karush-Kuhn-Tucker (KKT) conditions

Karush-Kuhn-Tucker (KKT) conditions

Consider the optimisation problem

$$\begin{aligned} \text{maximize } & f(x) \\ \text{subject to } & g_i(x) \leq 0 \\ & h_i(x) = 0 \end{aligned}$$

Form the Lagrangian

$$L(\boldsymbol{x}, \mu, \lambda) = f(\boldsymbol{x}) + \mu^T \boldsymbol{g}(\boldsymbol{x}) + \lambda^T \boldsymbol{h}(\boldsymbol{x})$$

Theorem 2

If $(\mathbf{x}^*, \mu^*)$ is a saddle point of $L(\mathbf{x}, \mu)$ in $\mathbf{x} \in \mathbf{X}$, $\mu \geq \mathbf{0}$, then $\mathbf{x}^*$ is an optimal vector for the above optimization problem. Suppose that $f(\mathbf{x})$ and $g_i(\mathbf{x})$, $i = 1, \ldots, m$, are convex in $\mathbf{x}$ and that there exists $\mathbf{x}_0 \in \mathbf{X}$ such that $\mathbf{g}(\mathbf{x}_0) < 0$. Then with an optimal vector $\mathbf{x}^*$ for the above optimization problem there is associated a non-negative vector μ^* such that $L(\mathbf{x}^*, \mu^*)$ is a saddle point of $L(\mathbf{x}, \mu)$

KKT conditions

$$\frac{\partial}{\partial w_\nu} L_P = w_\nu - \sum_i^n \lambda_i y_i x_{i\nu} = 0 \qquad \nu = 1, \ldots, p$$

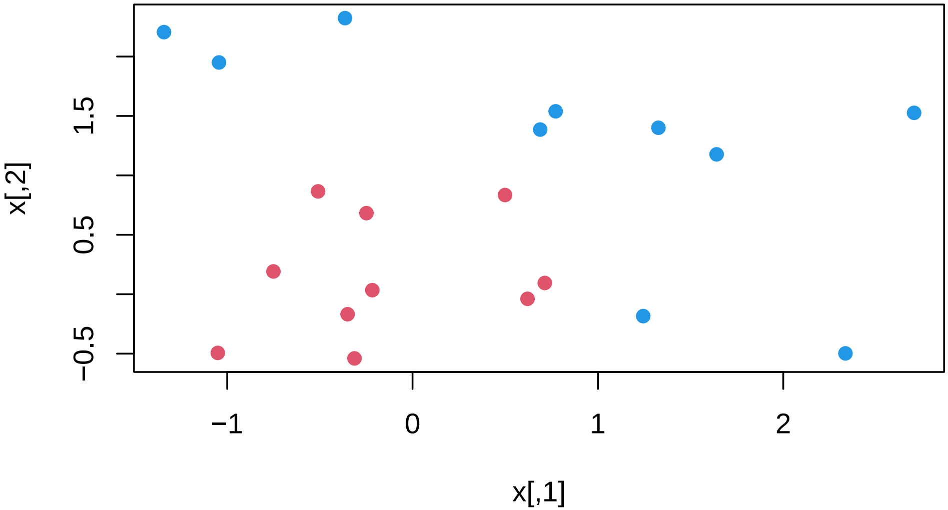
$$\frac{\partial}{\partial b} L_P = -\sum_{i=1}^n \lambda_i y_i = 0$$

$$\begin{aligned} y_i(\boldsymbol{x}_i^T \boldsymbol{w} + b) - 1 &\geq 0 \qquad i = 1, \ldots, n \\ \lambda_i &\geq 0 \qquad i = 1, \ldots, n \\ \lambda_i(y_i(\boldsymbol{x}_i^T \boldsymbol{w} + b) - 1) &= 0 \qquad i = 1, \ldots, n \end{aligned}$$

Numerical example

Example from here

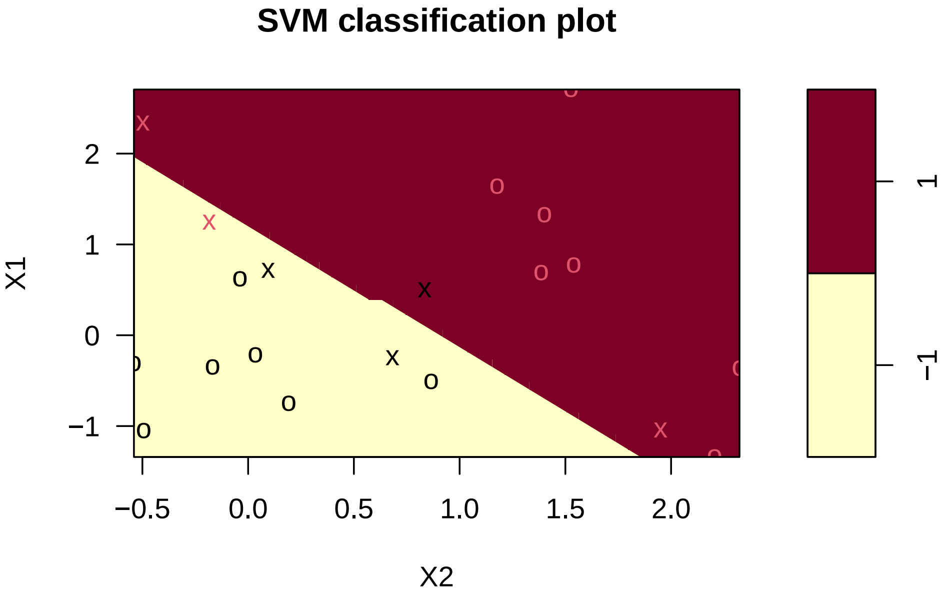
```
set.seed(10111)
x = matrix(rnorm(40), 20, 2)
y = rep(c(-1, 1), c(10, 10))
x[y == 1,] = x[y == 1,] + 1
plot(x, col = y + 3, pch = 19)
```


x[,1]
x[,2]
−1
0
1
2
−0.5
0.5
1.5

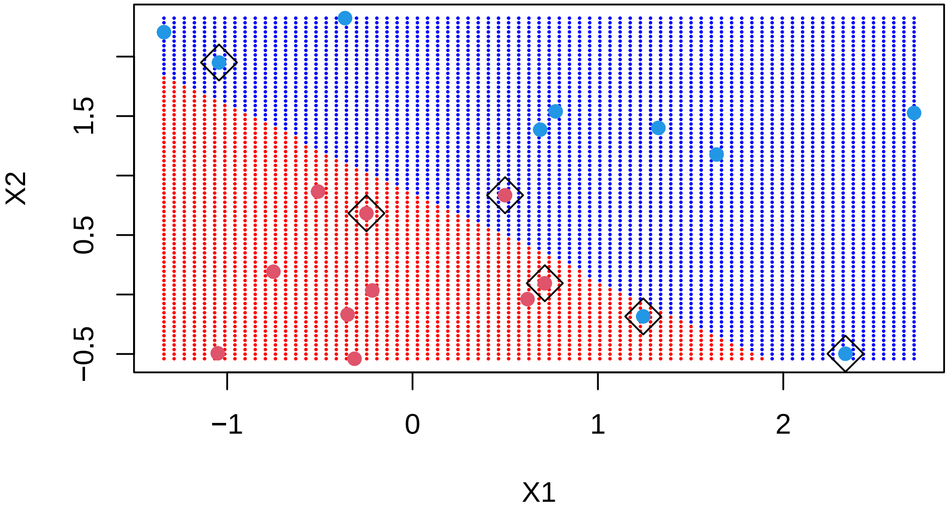
```
dat = data.frame(x, y = as.factor(y))
svmfit = svm(y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
print(svmfit)

##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10, scale = FA
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  6

plot(svmfit, dat)
```

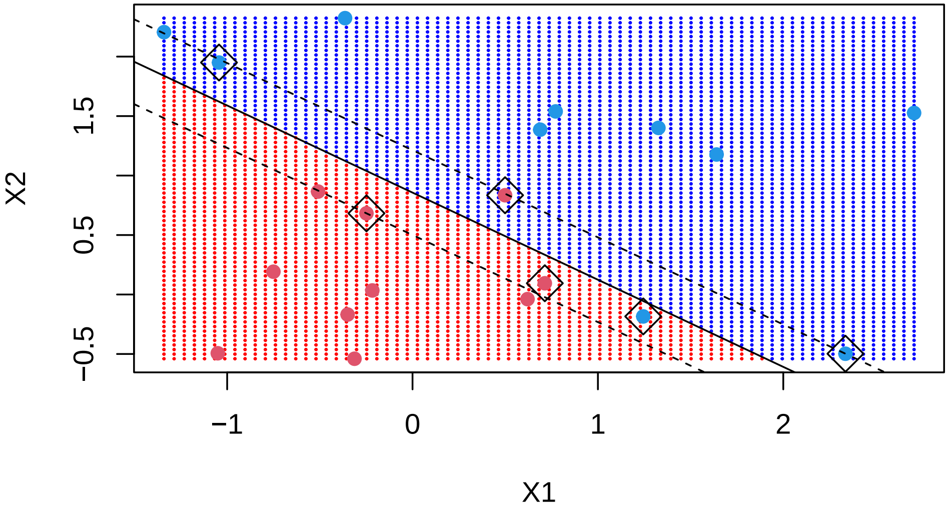

SVM classification plot
X1
X2
2
1
0
−1
−0.5
0.0
0.5
1.0
1.5
2.0
1
−1

```r
make.grid = function(x, n = 75) {
  grange = apply(x, 2, range)
  x1 = seq(from = grange[1,1], to = grange[2,1], length = n)
  x2 = seq(from = grange[1,2], to = grange[2,2], length = n)
  expand.grid(X1 = x1, X2 = x2)
}
xgrid = make.grid(x)
ygrid = predict(svmfit, xgrid)
plot(xgrid, col = c("red","blue")[as.numeric(ygrid)], pch = 20, cex = .2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index,], pch = 5, cex = 2)
```



X2
X1
−0.5
0.5
1.5
−1
0
1
2

```
beta = drop(t(svmfit$coefs)%*%x[svmfit$index,])
beta0 = svmfit$rho
plot(xgrid, col = c("red", "blue")[as.numeric(ygrid)], pch = 20, cex = .2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index,], pch = 5, cex = 2)
abline(beta0 / beta[2], -beta[1] / beta[2])
abline((beta0 - 1) / beta[2], -beta[1] / beta[2], lty = 2)
abline((beta0 + 1) / beta[2], -beta[1] / beta[2], lty = 2)
```



−0.5
0.5
1.5
X2
−1
0
1
2
X1

Soft-margin SVM

To extend SVM to cases in which the data are not linearly separable, the **hinge loss** function is helpful

$$\max\left(0, 1 - y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b)\right)$$

y_i is the ith target (i.e., in this case, 1 or -1), and $\mathbf{w}^\mathsf{T}\mathbf{x}_i - b$ is the i-th output

This function is zero if the constraint is satisfied, in other words, if $\mathbf{x}_i$ lies on the correct side of the margin

For data on the wrong side of the margin, the function's value is proportional to the distance from the margin

The goal of the optimization then is to minimize

$$\lambda\|\mathbf{w}\|^2 + \left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b)\right)\right]$$

where the parameter $\lambda > 0$ determines the trade-off between increasing the margin size and ensuring that the $\mathbf{x}_i$ lie on the correct side of the margin

Thus, for sufficiently small values of λ, it will behave similar to the hard-margin SVM, if the input data are linearly classifiable, but will still learn if a classification rule is viable or not