SVM

Fernando Anorve

4/15/2020

Máquinas de Soporte Vectorial

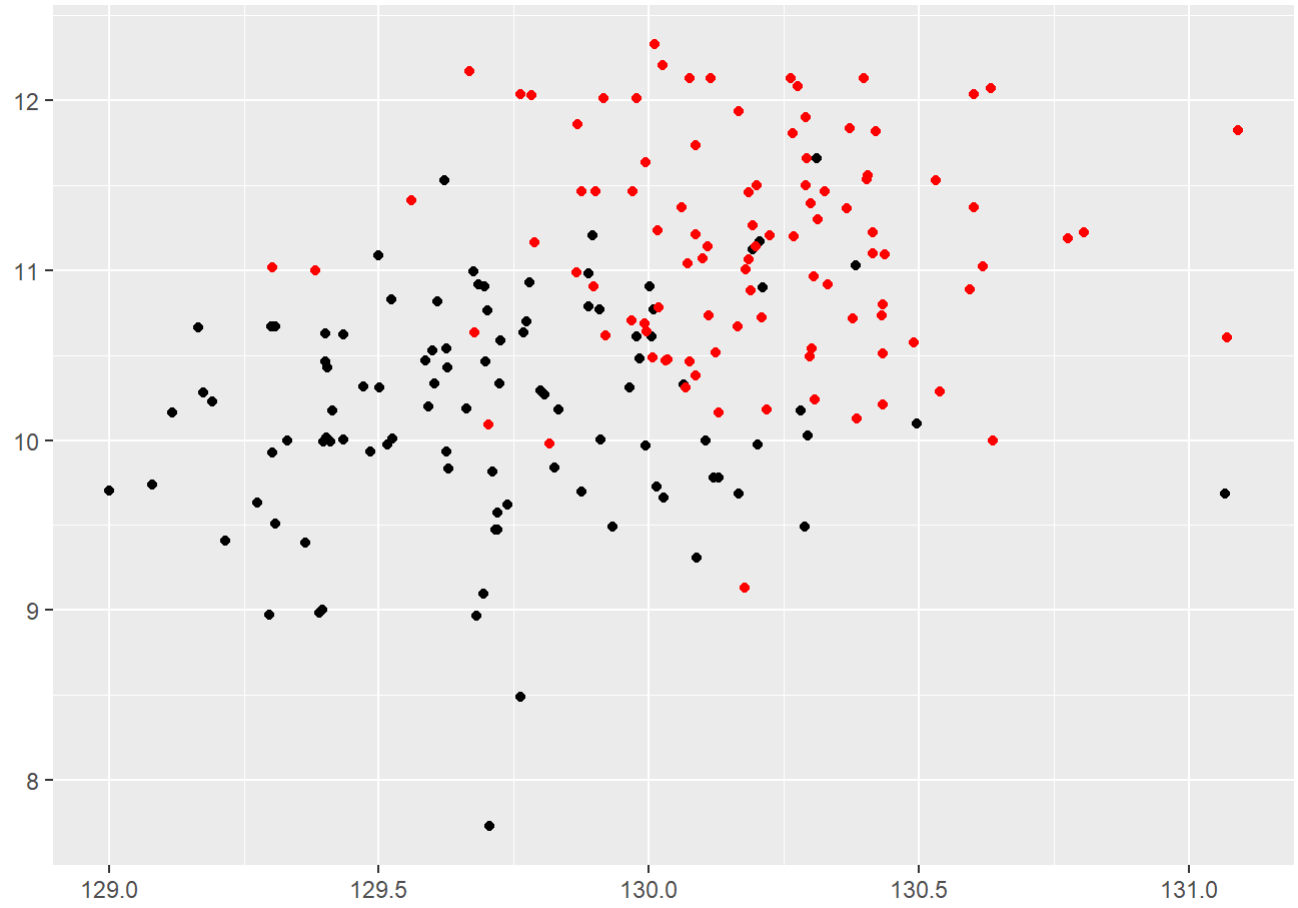
Clasificación por medio de un hiperplano

En nuestro problema de clasificación tenemos parejas de observaciones

$$(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n),$$

donde $X_i = (x_{1i}, x_{2i}, \ldots, x_{pi})^T \in \mathbb{R}^p$ y $y_i \in \{-1, 1\}$

Para entender esta técnica, es importante tener en cuenta que las variables de predicción X_i se hallan en un espacio $\mathbb{R}^p$, de p dimensiones. Por ejemplo, si $p = 2$, podríamos observar algo así:



¿Y si pudiéramos dividir el espacio en dos? De tal forma que las observaciones de un lado las predecimos como -1 y las observaciones del otro lado las predecimos como 1. ¡Para dividir el espacio podríamos usar un hiperplano!

Consideremos la función $f : \mathbb{R}^p \to \mathbb{R}$,

$$f(x) = x^T\beta + \beta_0$$

Una función así nos sirve para varias cosas:

1. El conjunto de puntos $\{x : f(x) = x^T\beta + \beta_0 = 0\}$ nos define un hiperplano que divide al espacio en dos partes
2. Las dos partes en las que divide el plano son $\{x : f(x) = x^T\beta + \beta_0 < 0\}$ y $\{x : f(x) = x^T\beta + \beta_0 > 0\}$. Evidentemente, la frontera entre ambas pre-imágenes es el hiperplano del punto anterior
3. Cuando $||\beta|| = 1$, $|f(x)|$ nos dice la distancia entre el punto x y el hiperplano

Estos tres puntos anteriores nos dan una pista sobre cómo podemos usar f para clasificar los puntos $X_1, X_2, \ldots, X_n$... ¡Usando su signo!

Usaremos la regla de clasificación $G(x) = \text{sign}(f(x))$. Por lo tanto deseamos una f de tal manera que en la medida de lo posible $y_i < 0 \implies y_i \in \{x : f(x) = x^T\beta + \beta_0 < 0\}$ y $y_i > 0 \implies y_i \in \{x : f(x) = x^T\beta + \beta_0 > 0\}$

En pocas palabras, es deseable que y_i y $f(x_i)$ tengan el mismo signo, id est, $y_i f(x_i) > 0$. Cuando esto es posible para toda $i \leq n$, decimos que el conjunto es separable. En decir, el hiperplano separa perfectamente a las y_i según su valor.

Problema de maximización (Caso separable)

Para hallar el hiperplano que buscamos, planteamos el problema de optimización

$$\max_{\beta_0, ||\beta||=1} M$$

sujeto a

$$y_i(x_i^T\beta + \beta_0) \geq M, i = 1, \ldots, N$$

Recordemos que $|x^T\beta + \beta_0|$ nos dice qué tan alejado está x del hiperplano y en particular $y_i(x_i^T\beta + \beta_0)$ dice qué tan alejado está x_i del “bando opuesto”.

- Cuando $y_i(x_i^T\beta + \beta_0) > 0$, y_i está donde le toca ir
- Cuando $y_i(x_i^T\beta + \beta_0) < 0$, y_i está en el bando opuesto

Por eso queremos maximizar M, para que $y_i(x_i^T\beta + \beta_0)$ sea lo más grande posible. M es el margen alrededor del hiperplano.

Problema de maximización (Caso no separable)

En caso no separable deja de ser tan intuitivo como el anterior. A grandes rasgos, $y_i(x_i^T\beta + \beta_0) < 0$ forzosamente será negativo en algunos casos, por lo que la condición $\geq M$ cambia.

- Después de incorporar ciertas variables de holgura $\xi_i \geq 0$ y algunas equivalencias del problema, obtendremos el problema de minimización:

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$

sujeto a $\xi_i \geq 0, y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \forall i$

En castellano, ξ_i representa la proporción que la observación x_i se "sale" de su lado del margen. Cuando $\xi = 0$, quiere decir que x_i está en la "zona segura", mientras que cuando x_i está ya en el lado opuesto al que corresponde (es decir, que $y_i f(x_i) < 0$) entonces $\xi_i > 1$.

Dicho de otra forma, entre más x_i estén en medio de los márgenes o de plano en el contrario al que predice $f(x)$, entonces $\sum_{i=1}^{N}\xi_i$ va a ser más pesada.

Entre más grande sea C, más va a pesar que los x_i entren en el margen, por lo que va a tender a ser más pequeño. Entre más pequeño sea C, menos va a importar que los x_i entren en el margen, por lo que va a tender a ser más grande.

Ejemplo

En un ejemplo muy particular que se cubrió en regresión logística se había mencionado el conjunto de datos "banknote" de la librería uskewFactors.

Este conjunto contiene medidas de los márgenes y las diagonales de 200 billetes de banco, donde 100 son falsos y 100 son genuinos. La variable respuesta en este conjunto está marcada como Y.

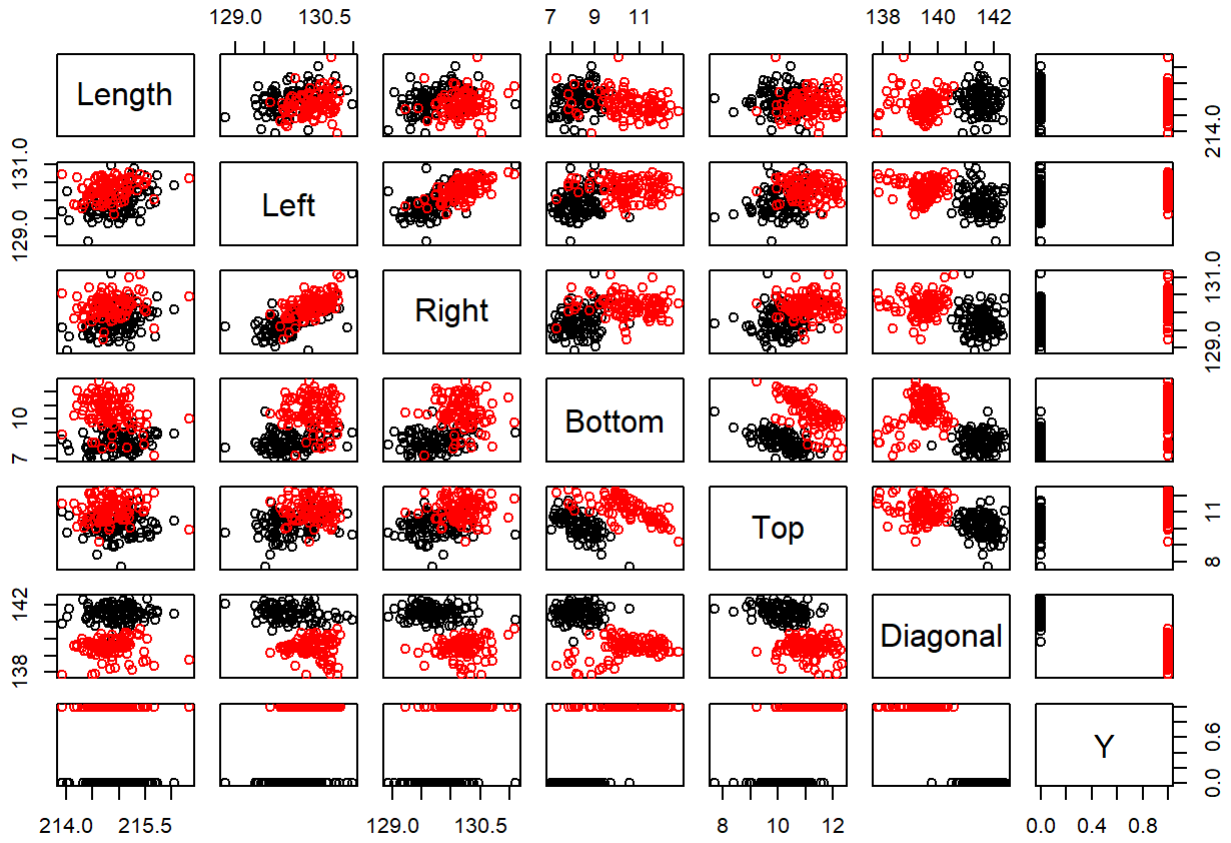
- Y = 0 cuando el billete de banco es falso *¨Y = 1 cuando el billete de banco es genuino

```
library(uskewFactors)

data(banknote)
names(banknote)
```

```
## [1] "Length"   "Left"     "Right"    "Bottom"   "Top"      "Diagonal"
## [7] "Y"
```

Puesto que este conjunto de datos es casi separable, la regresión logística no puede llevarse a cabo.

```
pairs(banknote, col = banknote$Y + 1)
```

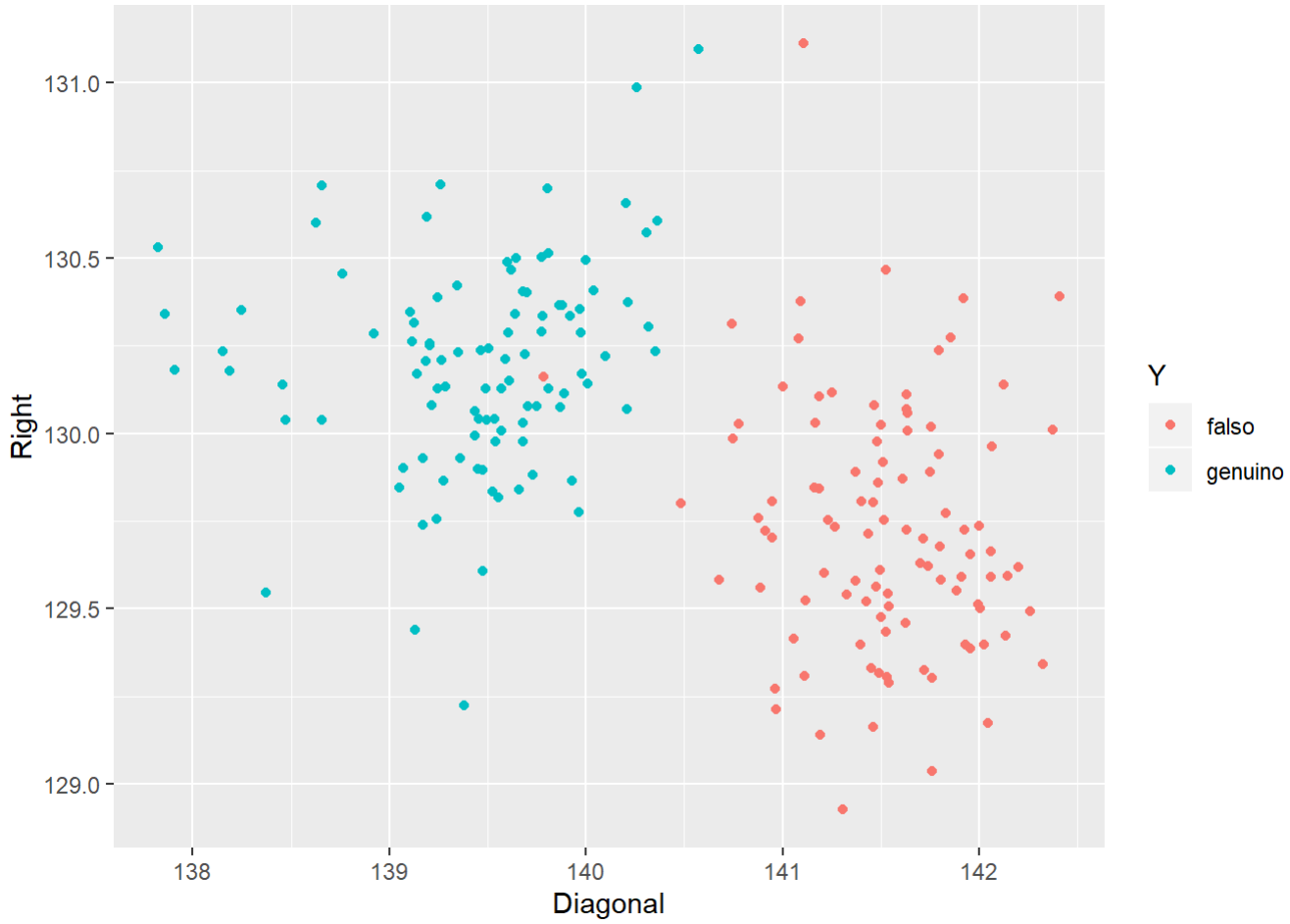



```
banknote$Y = as.factor(ifelse(banknote$Y == 0,"falso","genuino"))
```

En cambio, podemos utilizar este conjunto para ver ejemplos de cómo implementar métodos de soporte vectorial en R.

Primero observamos que si reducimos las variables predictivas a la diagonal y al márgen derecho, el problema sigue siendo separable

```
library(ggplot2)
ggplot(data = banknote , aes(x = Diagonal, y = Right,color = Y)) + geom_point()
```



Esta es una buena oportunidad para implementar clasificación por soporte vectorial

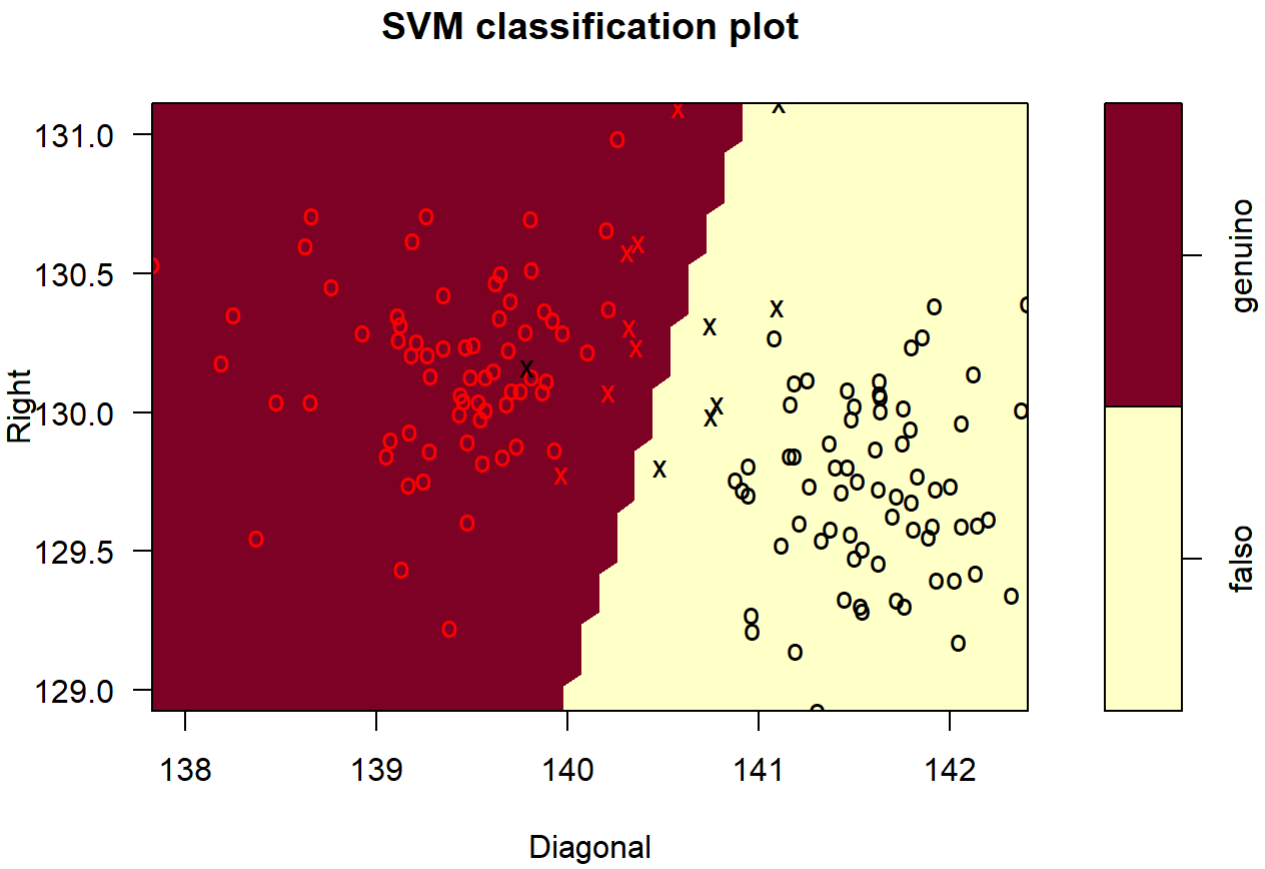
La librería *e1071* contiene implementaciones de algunas técnicas de aprendizaje estadístico, y entre ellas se encuentra la función svm() que nos interesa en este momento

- Cuando especificamos que el kernel que usaremos es lineal podemos usar la función para ajustar un clasificador de soprte vectorial
- El parámetro *cost* especifica el costo de una infracción a los márgenes
    - Cuando el costo es pequeño, hay más oportunidad para que el margen sea grande, y que haya más puntos que lo infrinjan
    - Cuando el costo es alto, el modelo es más restrictivo y por lo tanto el margen es más estrecho

Visualización (forma directa)

El resultado de la función svm ya tiene un método “plot” que sirve para visualizar el fenómeno en una gráfica de dos dimensiones. No obstante, no necesariamente es tan claro como nos gustaría

```
set.seed(2020)
trainset = sample(200,150,replace = F)

banknote.train = banknote[trainset,]
banknote.test = banknote[-trainset,]

attach(banknote.train)

library(e1071)
svmfit = svm(Y ~ Right + Diagonal , kernel = "linear", cost = 1 , scale = FALSE)

plot(svmfit, data.frame(Y , Right, Diagonal))
```



Visualización (“a mano”)

Otra forma de visualizar la gráfica es calculando los parámetros pendiente-ordenada a mano a partir del resultado de svm()

1. Se calcula un vector w

```
w <- t(svmfit$coefs) %*% svmfit$SV
```

2. El vector w se utiliza para hallar la pendiente

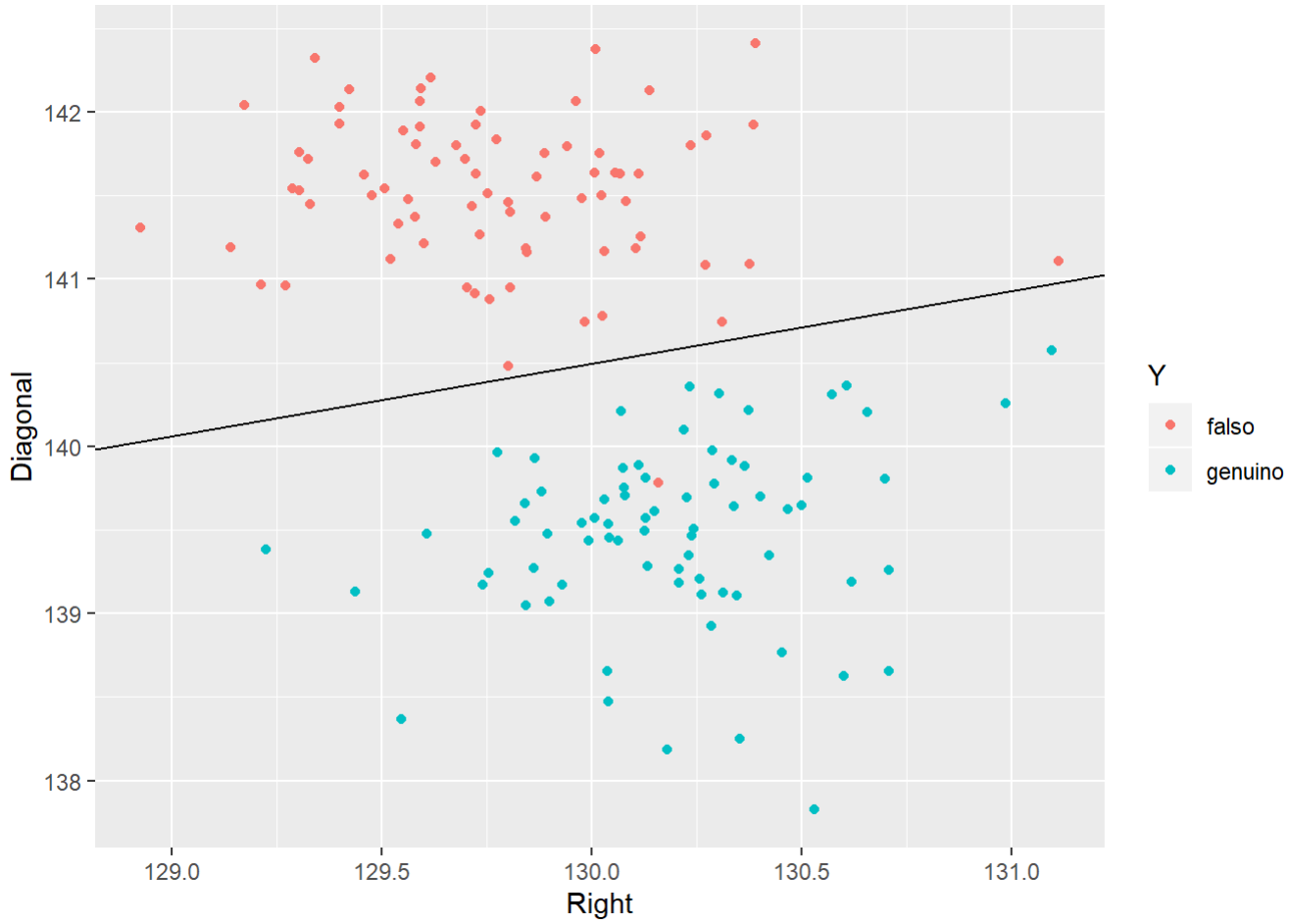
```
slope_1 <- -w[1]/w[2]
```

3. Usando la variable rho y el vector w se calcula la ordenada al origen

```
intercept_1 <- svmfit$rho/w[2]
```

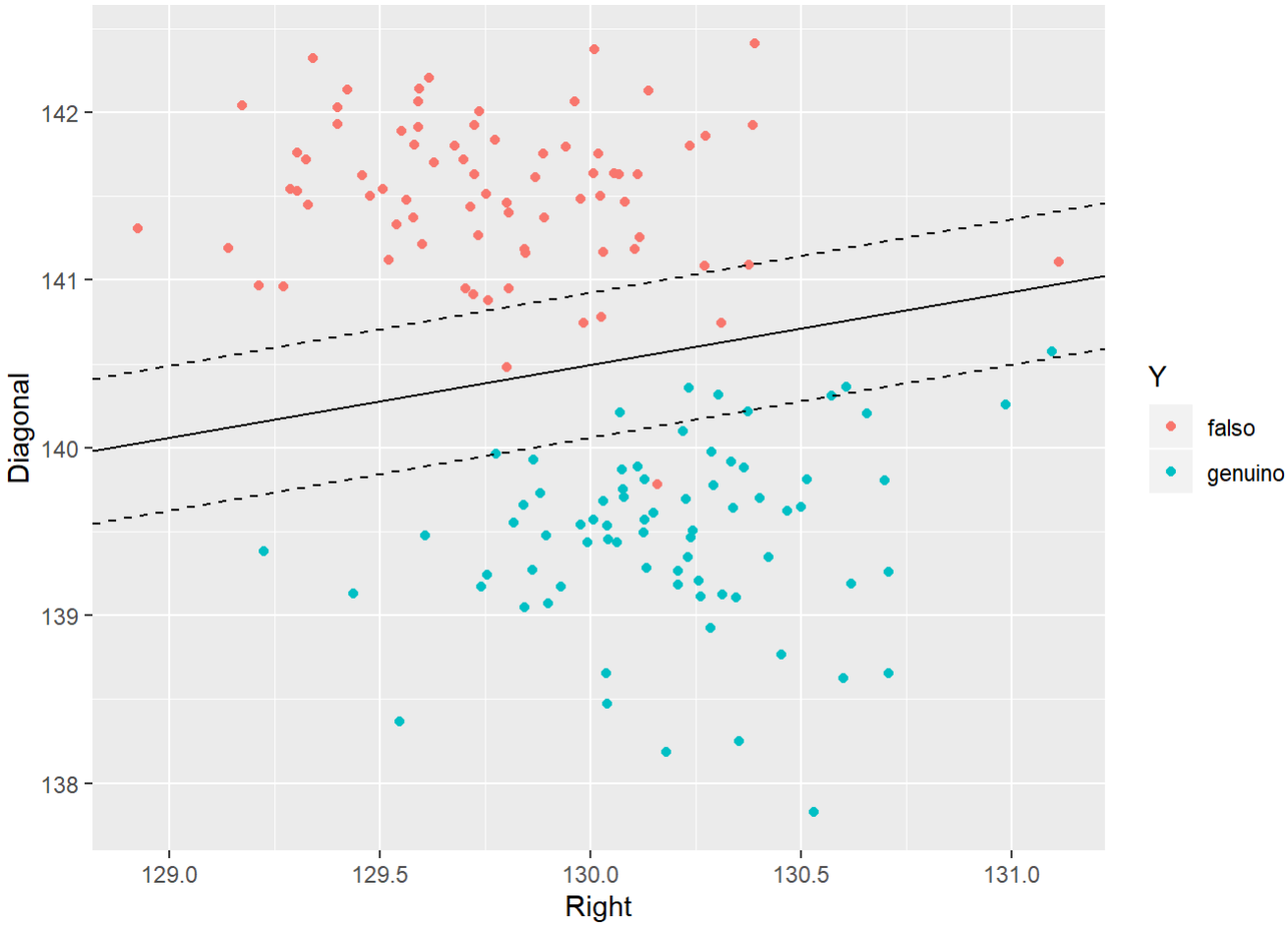
Habiendo calculado los parámetros de la recta,

```
p <- ggplot(data = banknote.train , aes(x = Right, y = Diagonal,color = Y)) + geom_point() +
  geom_abline(slope = slope_1,intercept = intercept_1)

p
```



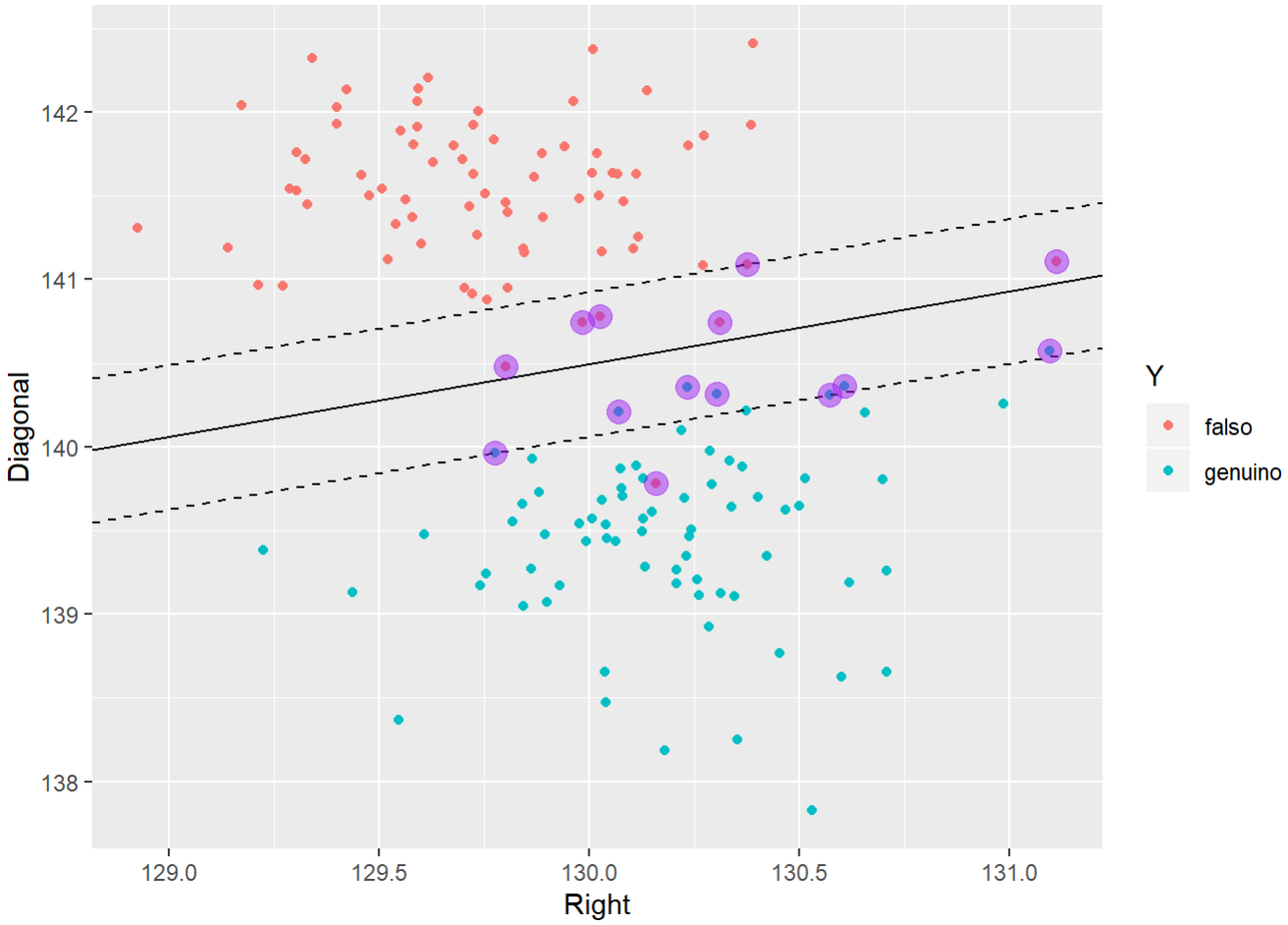
No hemos puesto todavía el margen, que se calcula sumandole al parámetro “intercept” +/- 1/w[2]

```
p <- p +
    geom_abline(slope = slope_1,
                intercept = intercept_1-1/w[2],
                linetype = "dashed") +
    geom_abline(slope = slope_1,
                intercept = intercept_1+1/w[2],
                linetype = "dashed")
p
```



¡Además es posible identificar los vectores de soporte!

```
df_sv <- banknote.train[svmfit$index,]
#mark out support vectors in plot
p <- p + geom_point(data = df_sv,
                    aes(x = Right, y = Diagonal),
                    color = "purple",
                    size = 4, alpha = 0.5)
p
```



Predicción del conjunto de prueba

```
svm.pred = predict(svmfit,banknote.test,type = "class")

conf_table = table(svm.pred,banknote.test$Y)

conf_table
```

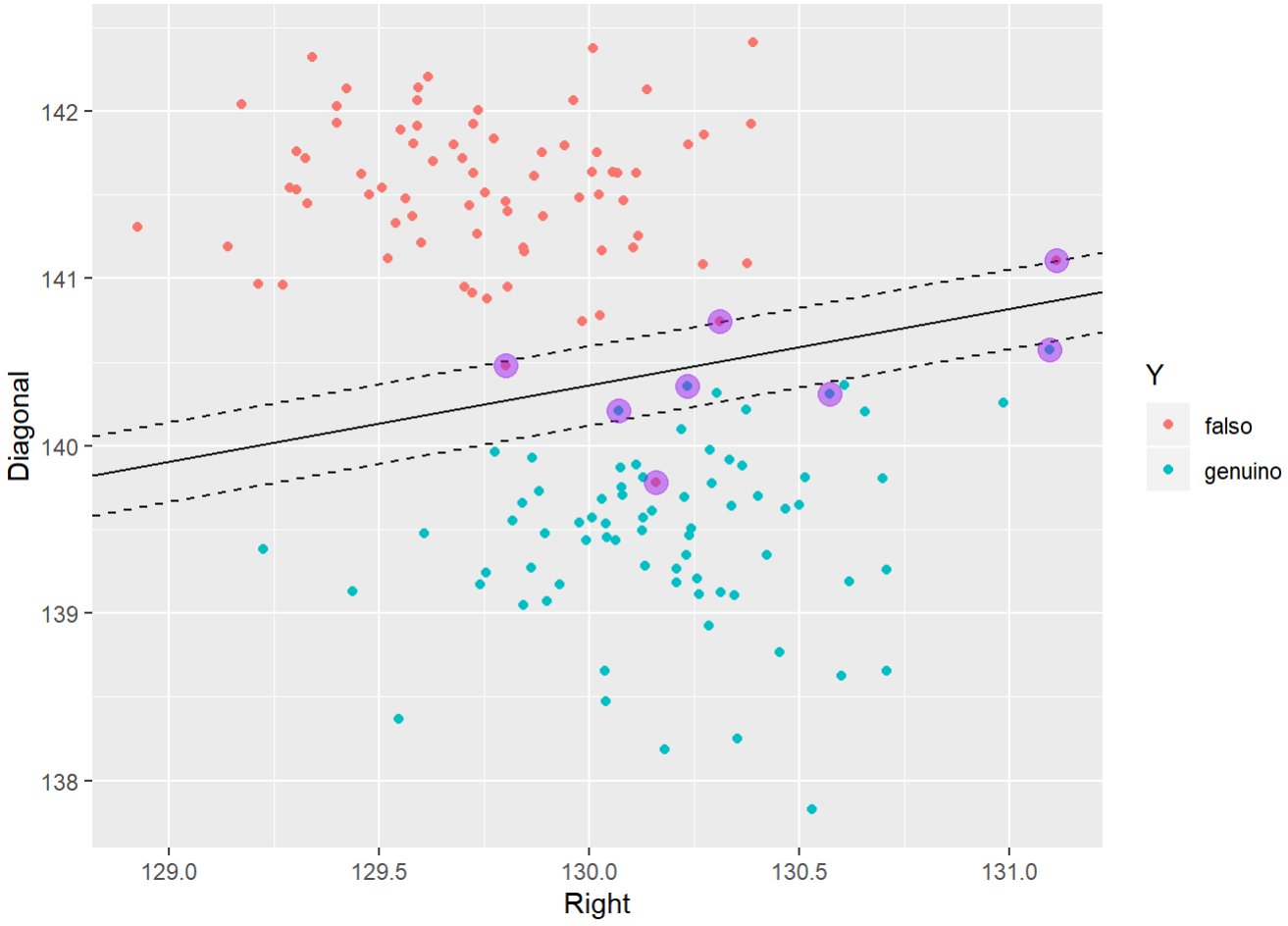
```
##
## svm.pred  falso genuino
##   falso      24       0
##   genuino     0      26
```

```
(conf_table[1,1] + conf_table[2,2])/sum(conf_table)
```

```
## [1] 1
```

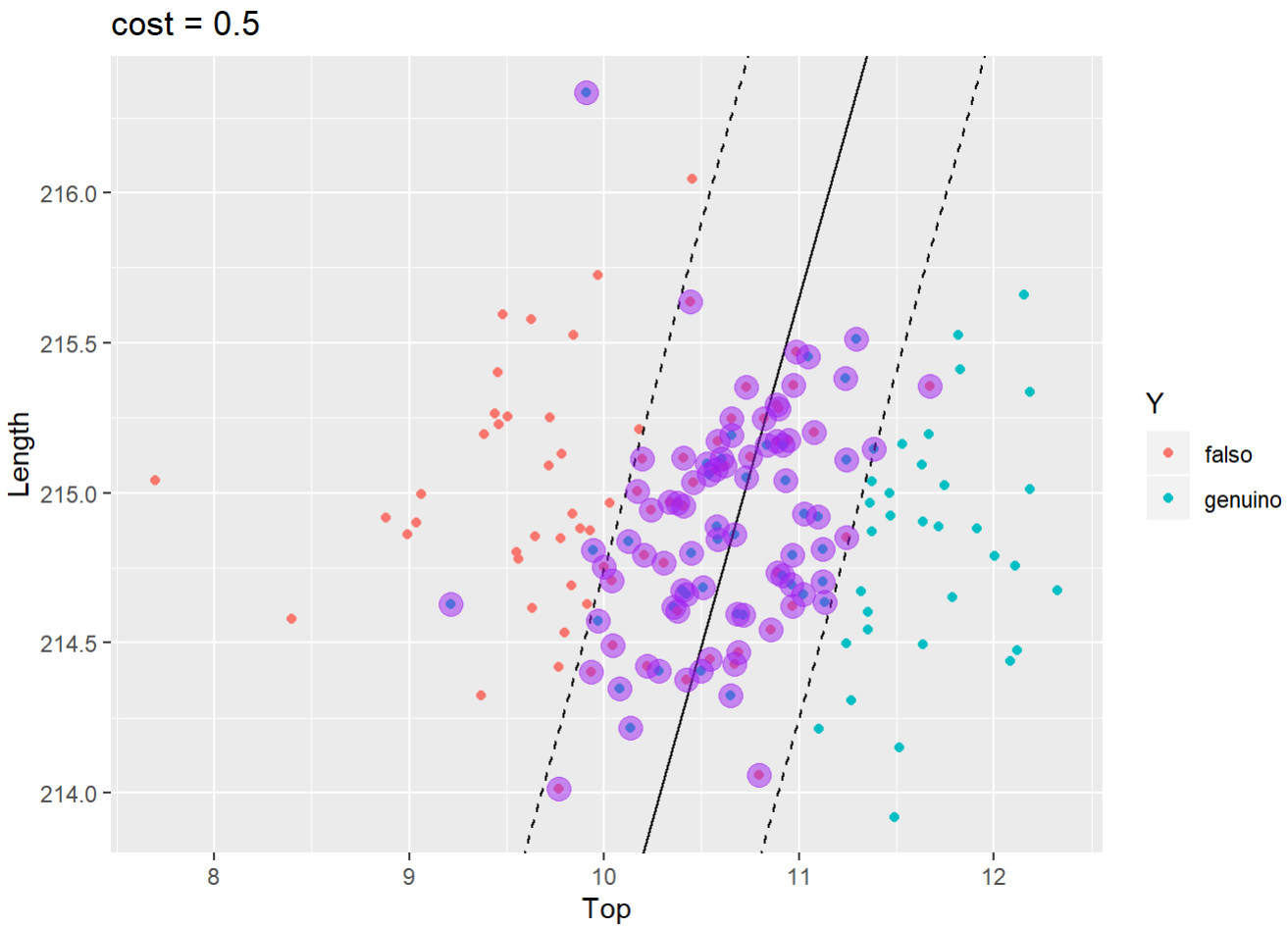
En cambio, cuando aumentamos el parámetro de costo,

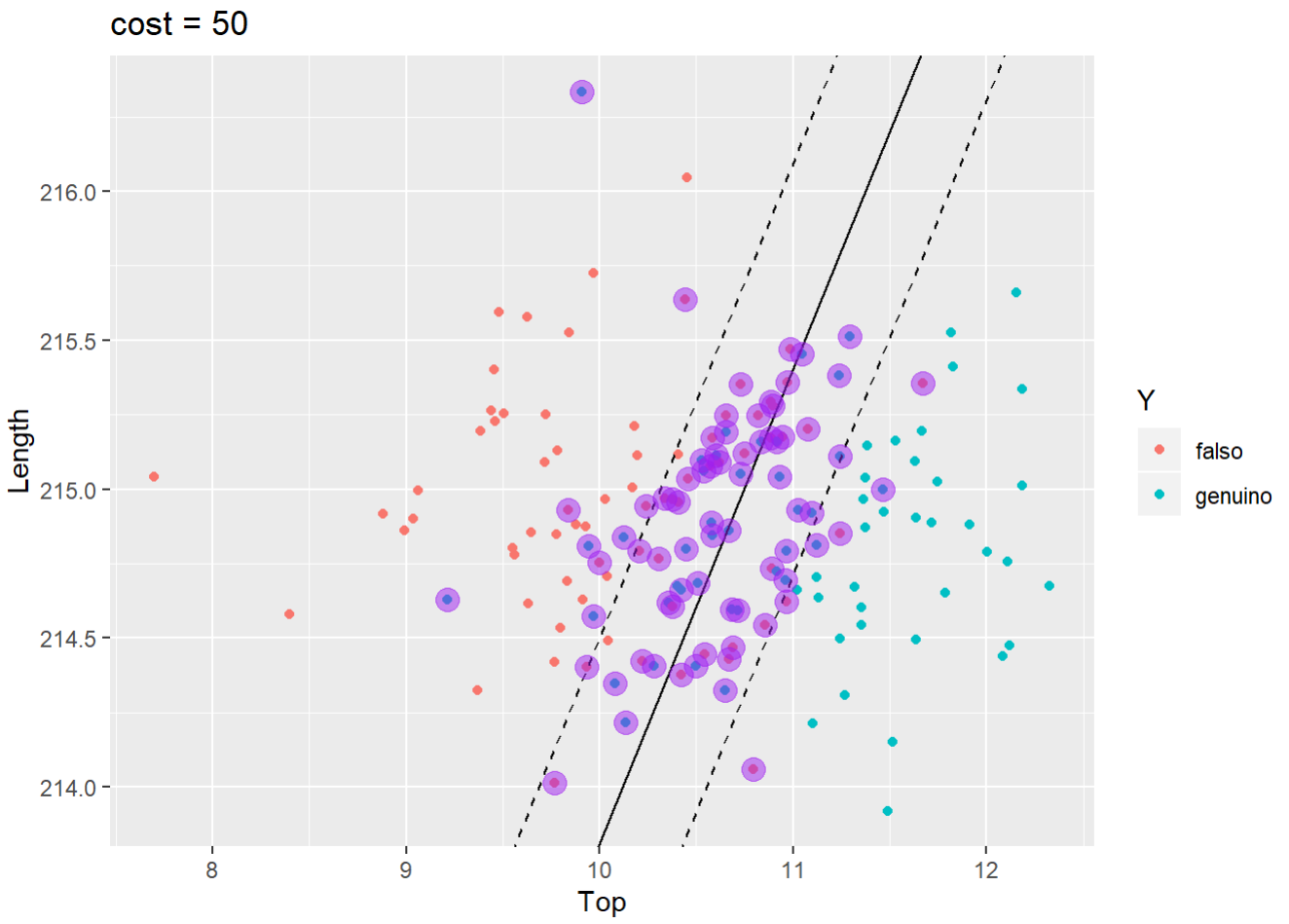
```
svmfit = svm(Y ~ Right + Diagonal , kernel = "linear", cost = 100 , scale = FALSE)

w <- t(svmfit$coefs) %*% svmfit$SV

slope_1 <- -w[1]/w[2]

intercept_1 <- svmfit$rho/w[2]

p <- ggplot(data = banknote.train , aes(x = Right, y = Diagonal,color = Y)) + geom_point() +
  geom_abline(slope = slope_1,intercept = intercept_1) +
    geom_abline(slope = slope_1,
                intercept = intercept_1-1/w[2],
                linetype = "dashed") +
    geom_abline(slope = slope_1,
                intercept = intercept_1+1/w[2],
                linetype = "dashed")

df_sv <- banknote.train[svmfit$index,]

p <- p + geom_point(data = df_sv,
                    aes(x = Right, y = Diagonal),
                    color = "purple",
                    size = 4, alpha = 0.5)
p
```



Los puntos rodeados de morado son el soporte vectorial. Son el soporte puesto que si los cambias a ellos, cambias la naturaleza de la recta y los márgenes.

Mismo ejemplo, con otro par de variables





Es de esperarse que la predicción no sea tan exacta como antes

```
svm.pred = predict(svmfit,banknote.test,type = "class")

conf_table = table(svm.pred,banknote.test$Y)

conf_table
```

```
##
## svm.pred  falso genuino
##   falso      18       3
##   genuino     6      23
```

```
(conf_table[1,1] + conf_table[2,2])/sum(conf_table)
```

```
## [1] 0.82
```

Mismo ejemplo, más variables

La función svm no sólo funciona con un par de variables. De hecho podemos usar todas las demás variables para predecir Y. La desventaja es que ya no es tan trivial mostrarlo en 2D. Aun así, podemos predecir el conjunto de prueba con bastante exactitud

```
detach(banknote.train)

trainset = sample(200,round(400/3),replace = F)

banknote.train = banknote[trainset,]
banknote.test = banknote[-trainset,]


svmfit = svm(Y ~ . , data = banknote.train, kernel = "linear", cost = 100 , scale = FALSE)


svm.pred = predict(svmfit,banknote.test,type = "class")

conf_table = table(svm.pred,banknote.test$Y)

conf_table
```

```
##
## svm.pred  falso genuino
##   falso      35       0
##   genuino     0      32
```

```
(conf_table[1,1] + conf_table[2,2])/sum(conf_table)
```

```
## [1] 1
```