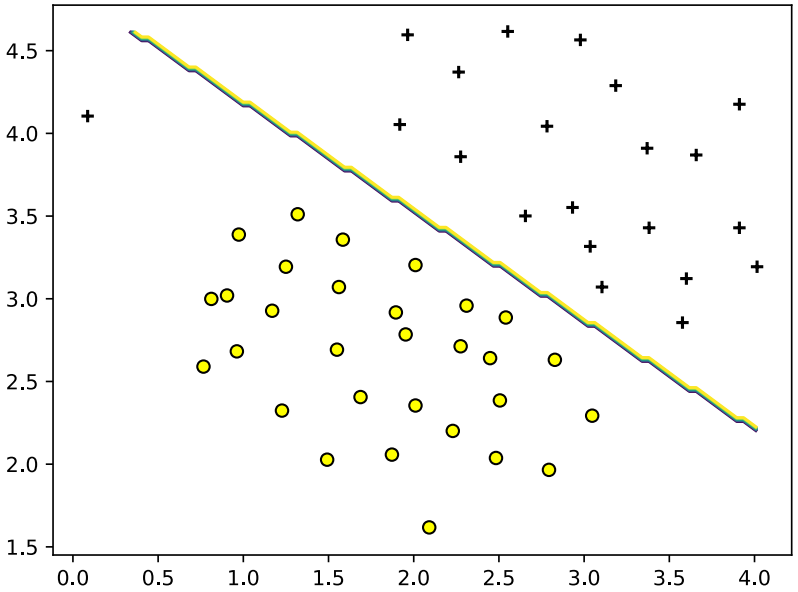
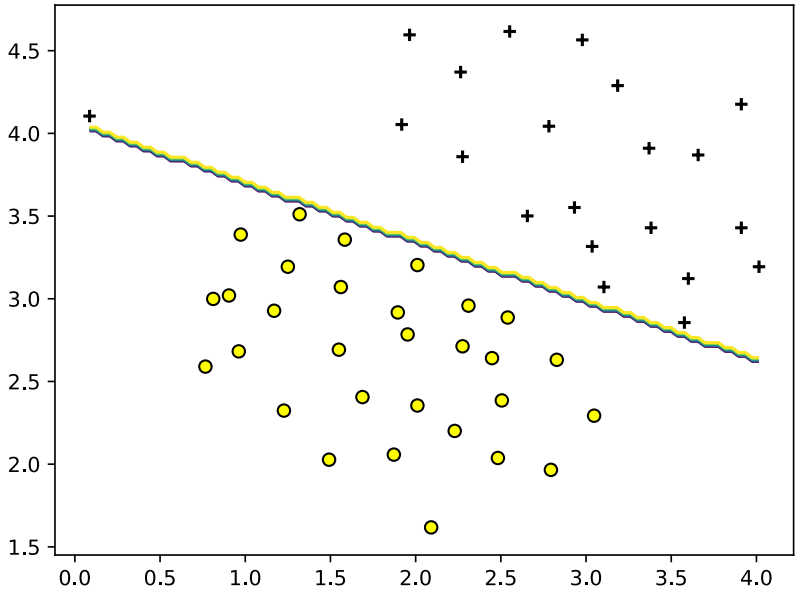
Comentarios sobre la práctica de support vector machines

Kernel lineal: selección de C



C = 1

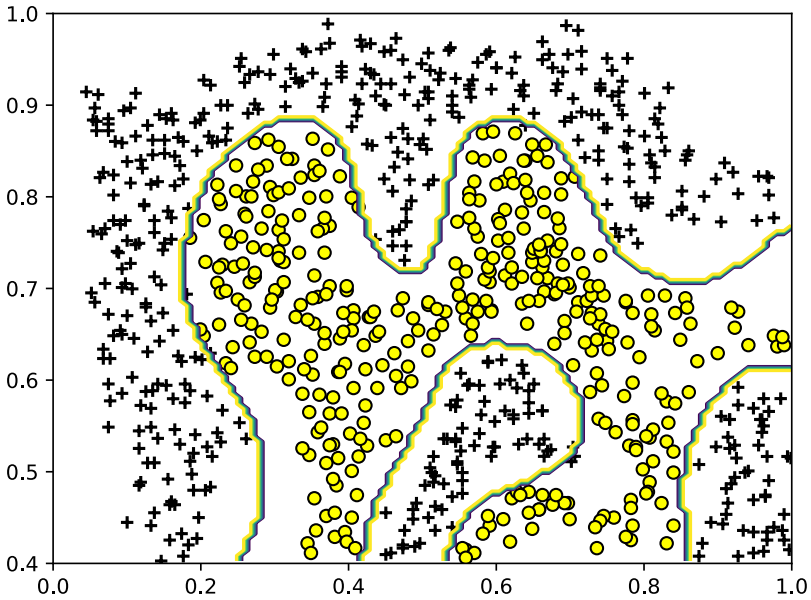


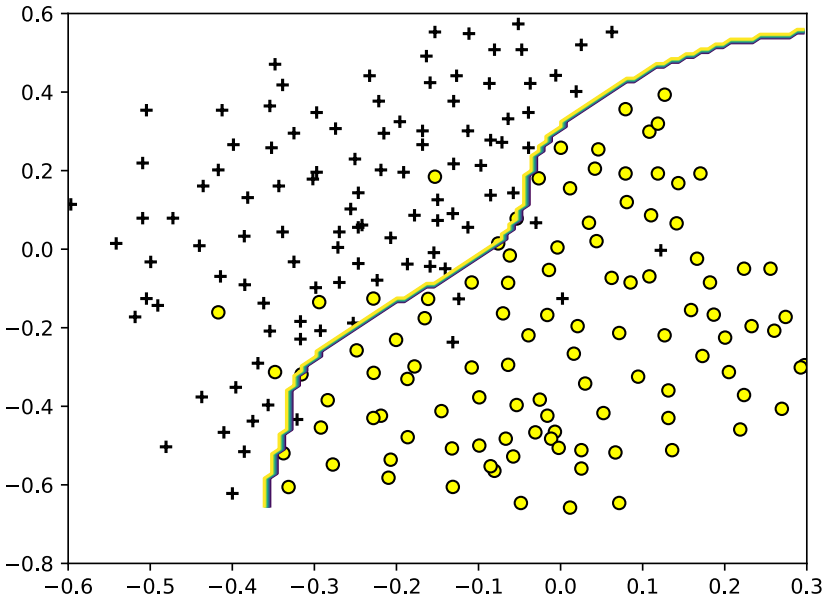
C = 100

```
def visualize_boundary(X, y, svm, file_name):
    x1 = np.linspace(X[:, 0].min(), X[:, 0].max(), 100)
    x2 = np.linspace(X[:, 1].min(), X[:, 1].max(), 100)
    x1, x2 = np.meshgrid(x1, x2)
    yp = svm.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape)

    pos = (y == 1).ravel()
    neg = (y == 0).ravel()
    plt.figure()
    plt.scatter(X[pos, 0], X[pos, 1], color='black', marker='+')
    plt.scatter(
        X[neg, 0], X[neg, 1], color='yellow', edgecolors='black', marker='o')
    plt.contour(x1, x2, yp)
    plt.savefig(file_name)
    plt.close()
```

Kernel Gaussiano: selección de C y σ






```
C_vec = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
sigma_vec = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
scores = np.zeros((len(C_vec), len(sigma_vec)))
```

hyperparameter tuning

Model selection

1. $h_\theta(x) = \theta_0 + \theta_1 x$ $\quad \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ $\quad \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$
   $\vdots$
10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$ $\quad \theta^{(10)} \rightarrow J_{cv}(\theta^{(10)})$

Pick the model with minimum cross validation error

$$\theta_0 + \theta_1 x_1 + \cdots + \theta_4 x^4$$

Estimate generalization error for test set $J_{test}(\theta^{(4)})$

Choosing the regularization parameter λ

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

1. Try $\lambda = 0$ $\quad \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
2. Try $\lambda = 0.01$ $\quad \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$
3. Try $\lambda = 0.02$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$

$\vdots$

12. Try $\lambda = 10$ $\quad \theta^{(12)} \rightarrow J_{cv}(\theta^{(12)})$

Pick (say) $\theta^{(5)}$. Test error: $J_{test}(\theta^{(5)})$

Detección de spam

Detección de spam

- Corpus:
  - spam.zip: 500 mensajes
  - easy_ham.zip: 2551 mensajes
  - hard_ham.zip: 250 mensajes

- Objetivo:
  Evaluar distintas configuraciones del sistema de aprendizaje, incluyendo qué mensajes se usan para entrenamiento y cuáles para evaluación

- Representación de los mensajes:

  mensaje y vocabulario → mensaje procesado → vector de 0s y 1s

$x^{(i)}$ es un vector de 1899 (palabras del vocabulario) componentes de 0s y 1s, ¿de dónde sacamos $y^{(i)}$?

Procesamiento de los mensajes

- Lectura de mensaje (el nombre de fichero se puede generar con `format`)

```
email_contents = open( 'spam/0001.txt', 'r' ).read()
```

```
...
Más líneas de cabecera
...
<CENTER>Save up to 70% on Life Insurance.</CENTER></FONT><FONT color=3D#ff=
0000
face=3D"Copperplate Gothic Bold" size=3D5 PTSIZE=3D"10">
<CENTER>Why Spend More Than You Have To?
<CENTER><FONT color=3D#ff0000 face=3D"Copperplate Gothic Bold" size=3D5 PT=
SIZE=3D"10">
<CENTER>Life Quote Savings
...
If you reside in any state which prohibits e-mail solicitations for insuran=
ce, please disregard this email.<BR></
FONT><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR=
><BR><BR><BR></FONT></P></CENTER></CENTER></TR></TBODY></TABLE></CENTER></=
CENTER></CENTER></CENTER></CENTER></BODY></HTML>
```

- Procesamiento del mensaje

```
tokens  = email2TokenList(email_contents)
```

```
['save', 'up', 'to', 'number', 'on', 'life', 'insur', 'whi','spend', 'more', 'than',
'you', 'have', 'to', 'life', 'quot', 'save',
...
'pleas', 'disregard', 'thi', 'email']
```

Procesamiento de corpus

- Corpus:
  - spam.zip: 500 mensajes
  - easy_ham.zip: 2551 mensajes
  - hard_ham.zip: 250 mensajes

- Objetivo:
  Evaluar distintas configuraciones del sistema de aprendizaje, incluyendo qué mensajes se usan para entrenamiento y cuáles para evaluación

```
directorio = "spam"
i = 1
email_contents = codecs.open(
            '{0}/{1:04d}.txt'.format(directorio, i), 'r',
            encoding='utf-8', errors='ignore').read()
```

Construcción de array por filas

```
In [15]: X = np.empty((0, 5))

In [20]: for i in range(3):
    ...:     X = np.vstack((X, np.ones(5)*i))
    ...:

In [21]: X
Out[21]:
array([[0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1.],
       [2., 2., 2., 2., 2.]])
```