Taller de Verano: 100 páginas de Machine Learning

Maikol Solís

10/1/24

Tabla de contenidos

1 Taller de verano

2 Cómo funciona el aprendizaje supervisado

Veremos el caso de las máquinas de soporte vectorial (SVM) para clasificación.

- **Paso #1: Cargar librerías**

```
import matplotlib.pyplot as plt

from sklearn import svm
from sklearn.datasets import make_blobs
from sklearn.inspection import DecisionBoundaryDisplay

from scipy.stats import distributions
from numpy import sum
import numpy as np
```

- **Paso #2: Crear datos**

Se crean 40 puntos usando la función `make_blobs`. Esta crea un conjunto de puntos separados en dos grupos.

```
X, y = make_blobs(n_samples=40, centers=2, random_state=6)
```

- **Paso #3: Crear el modelo**

```
clf = svm.SVC(kernel="linear", C=1000)
```

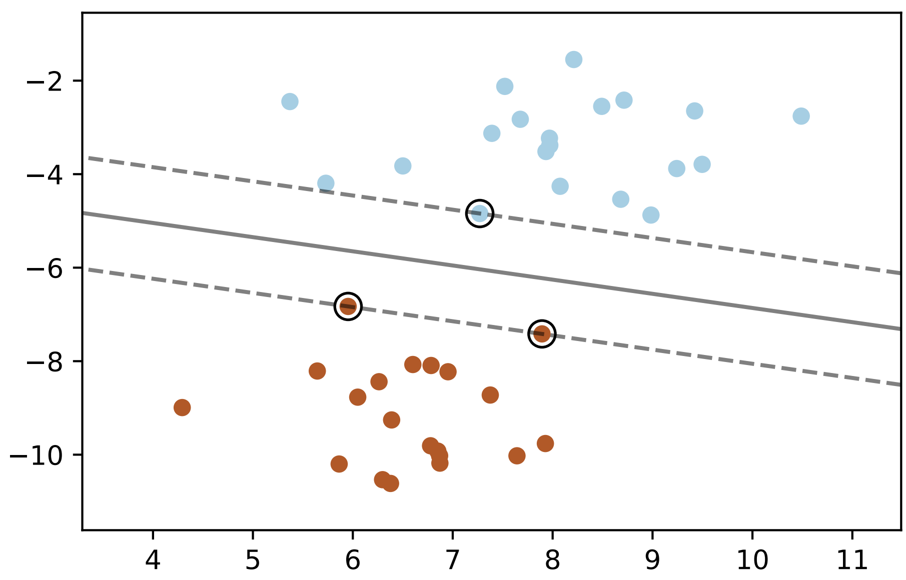
- **Paso #4: Entrenar el modelo**

```
clf.fit(X, y)
```

```
SVC(C=1000, kernel='linear')
```

- **Paso #5: Visualizar el modelo**

```python
plt.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Paired)

# plot the decision function
ax = plt.gca()
DecisionBoundaryDisplay.from_estimator(
    clf,
    X,
    plot_method="contour",
    colors="k",
    levels=[-1, 0, 1],
    alpha=0.5,
    linestyles=["--", "-", "--"],
    ax=ax,
)
# plot support vectors
ax.scatter(
    clf.support_vectors_[:, 0],
    clf.support_vectors_[:, 1],
    s=100,
    linewidth=1,
    facecolors="none",
    edgecolors="k",
)
plt.show()
```

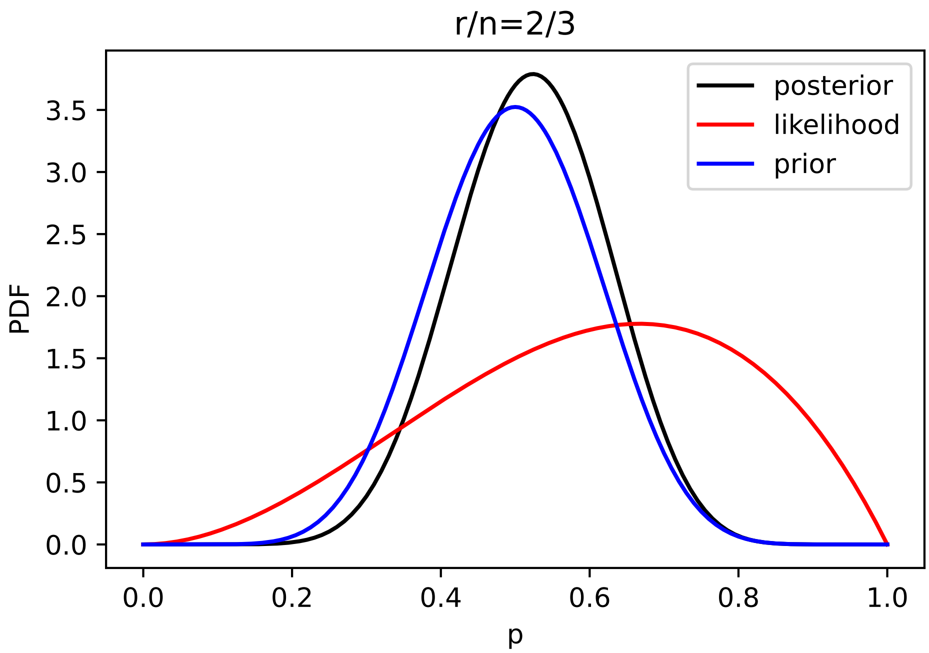


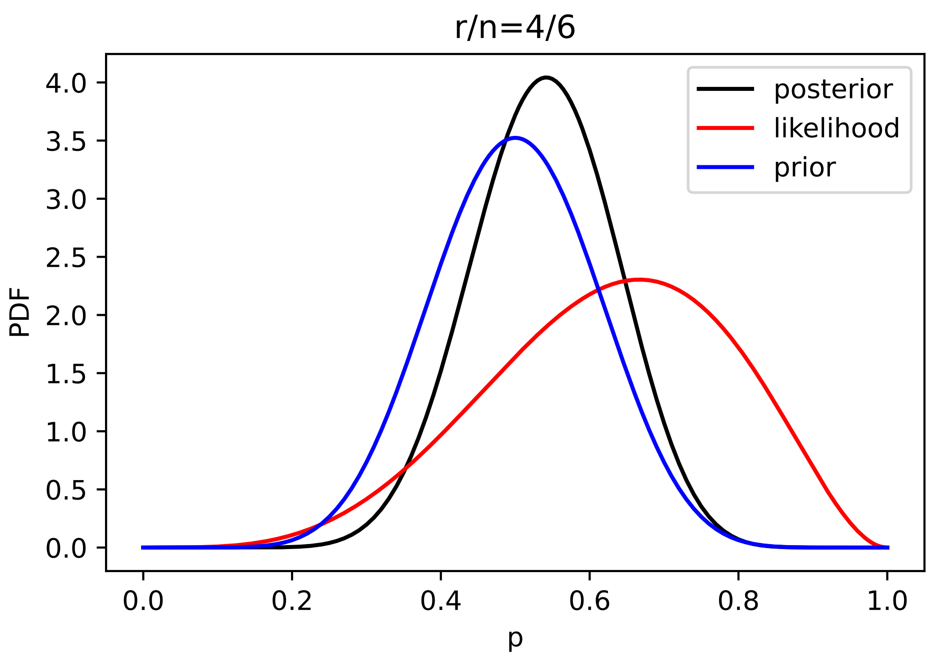
- **Referencias**

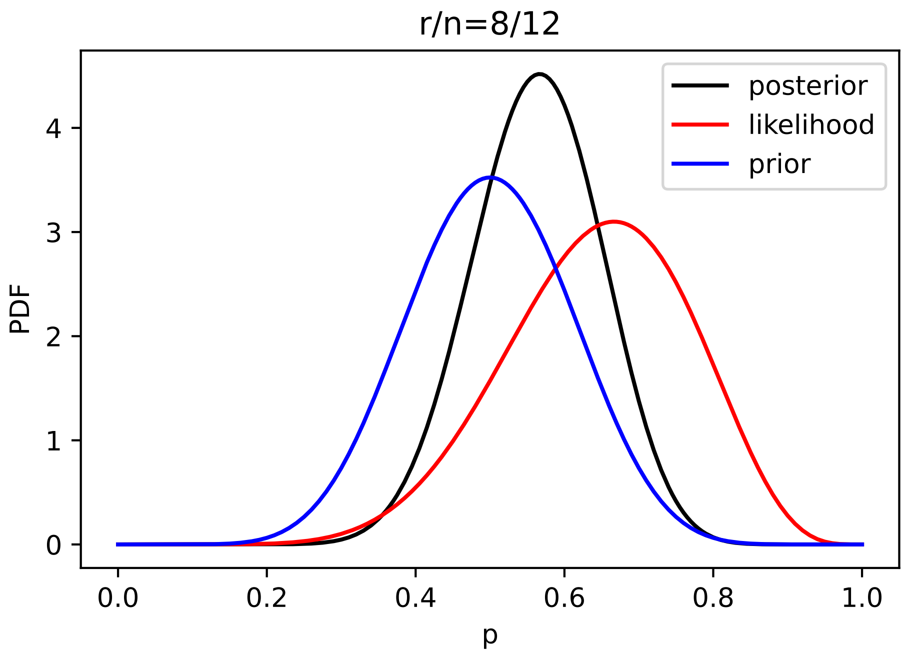
1. https://scikit-learn.org/stable/modules/svm.html#
2. https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html#sphx-glr-auto-examples-svm-plot-separating-hyperplane-py

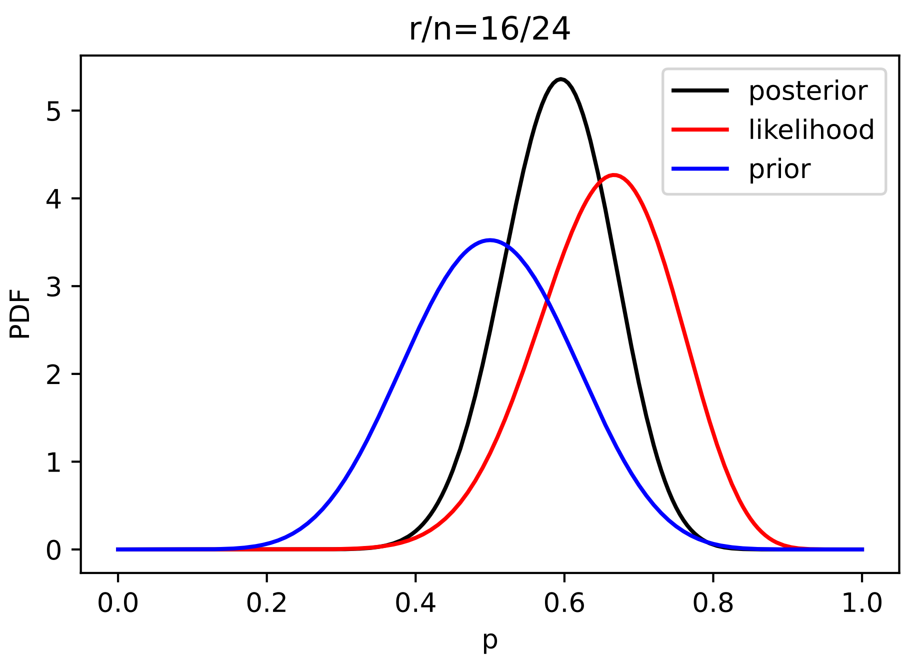
3 Estimación de parametros bayesiano

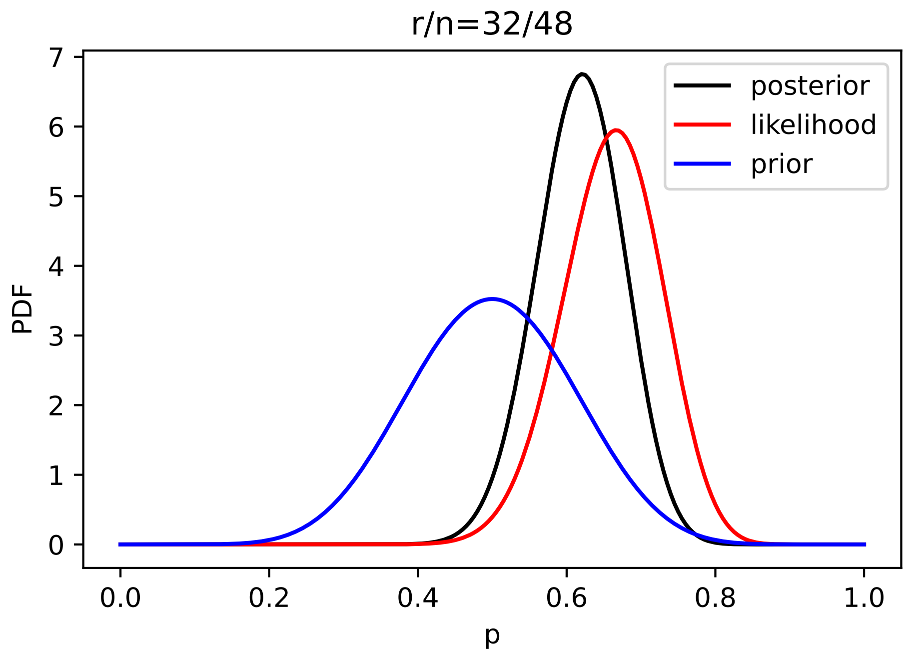
```
alpha = 10
beta = 10
n = 20
Nsamp = 201  # no of points to sample at
p = np.linspace(0, 1, Nsamp)
deltap = 1./(Nsamp-1)  # step size between samples of p

prior = distributions.beta.pdf(p, alpha, beta)

for i in range(1, 9):

    r = 2**i
    n = (3.0/2.0)*r
    like = distributions.binom.pmf(r, n, p)
    like = like/(deltap*sum(like))  # for plotting convenience only
    post = distributions.beta.pdf(p, alpha+r, beta+n-r)

    # make the figure
    plt.figure()
    plt.plot(p, post, 'k', label='posterior')
    plt.plot(p, like, 'r', label='likelihood')
    plt.plot(p, prior, 'b', label='prior')
    plt.xlabel('p')
    plt.ylabel('PDF')
    plt.legend(loc='best')
    plt.title('r/n={}/{:.0f}'.format(r, n))
    plt.show()
```

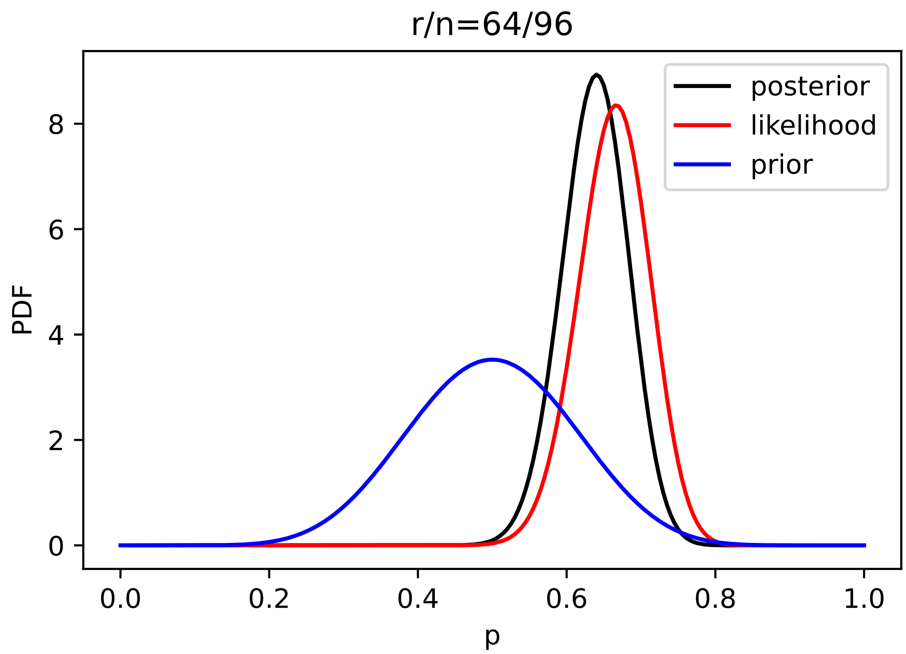

r/n=2/3
posterior
likelihood
prior
PDF
p
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.5
2.0
2.5
3.0
3.5

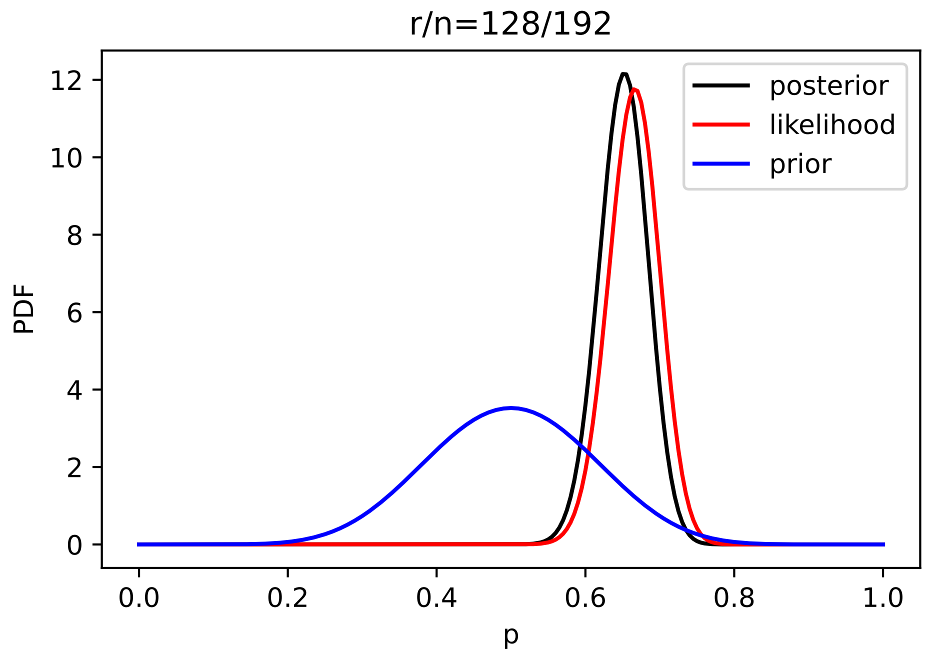

r/n=4/6
posterior
likelihood
prior
PDF
p
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.5
2.0
2.5
3.0
3.5
4.0


r/n=8/12
posterior
likelihood
prior
PDF
p
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0

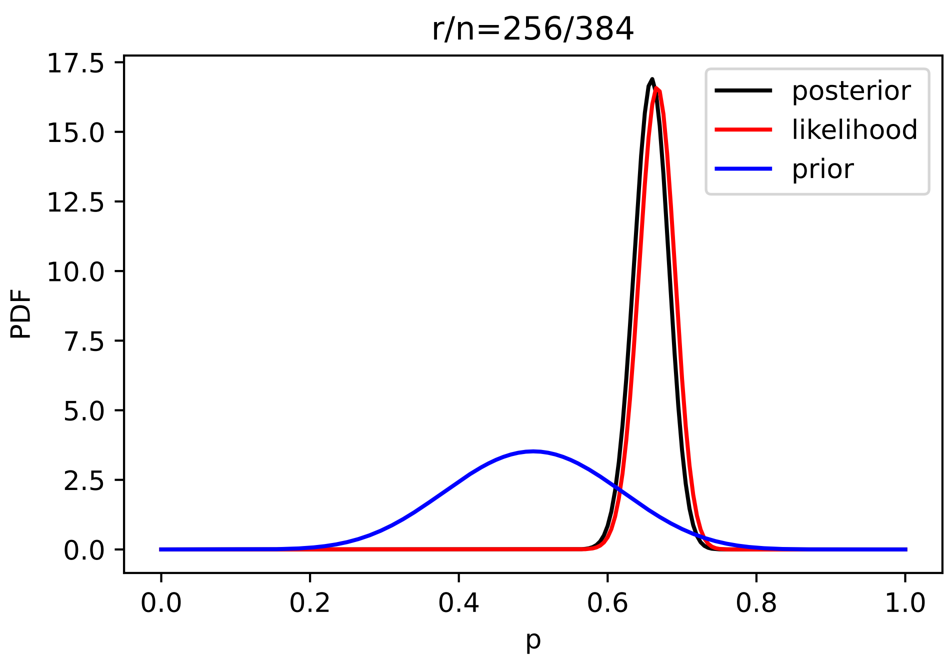

r/n=16/24
posterior
likelihood
prior
PDF
p
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0


r/n=32/48
posterior
likelihood
prior
PDF
p
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7


r/n=64/96
posterior
likelihood
prior
PDF
p
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8



r/n=128/192
posterior
likelihood
prior
PDF
p



r/n=256/384
posterior
likelihood
prior
PDF
p

4 Día #2

4.1 Regresión Lineal

4.2 Regresión Logística