

T2.5 MLE, minimización del riesgo empírico y regularización

Índice

1 Estimación máximo-verosímil

1.1 Definición

1.1.1 Estimador máximo-verosímil (MLE)

1.1.2 MLE con la conjunta

1.2 MLE para la Bernoulli (incondicional)

1.3 MLE para la categórica (incondicional)

1.4 MLE para la Gaussiana univariada (incondicional)

1.5 MLE para la Gaussiana multivariada (incondicional)

1.6 MLE para regresión (homocedástica) lineal (condicional)

2 Minimización del riesgo empírico

2.1 Definición y ejemplo: minimizar el error de clasificación

2.1.1 Definición

2.1.2 Ejemplo: minimizar el error de clasificación

2.2 Pérdida subrogada

2.2.1 Caso binario

3 Regularización

3.1 Definición

3.1.1 Regularización

3.1.2 Estimación maximum a posteriori (MAP)

3.2 Elección del regularizador mediante validación

3.3 Validación cruzada

3.3.1 La regla "un error estándar"

1 Estimación máximo-verosímil

1.1 Definición

1.1.1 Estimador máximo-verosímil (MLE)

Estimador máximo-verosímil (MLE): de un vector de parámetros θ con respecto a un conjunto de N datos $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}$ independientes e idénticamente distribuidos según una fdp (o fp) $p(\mathbf{y} \mid \mathbf{x}, \theta)$

$$\begin{aligned}\hat{\theta}_{\text{mle}} &= \underset{\theta}{\operatorname{argmax}} L(\theta) \quad \text{con} \quad L(\theta) = p(\mathcal{D} \mid \theta) = \prod_{n=1}^N p(\mathbf{y}_n \mid \mathbf{x}_n, \theta) \\ &= \underset{\theta}{\operatorname{argmax}} LL(\theta) \quad \text{con} \quad LL(\theta) = \log L(\theta) = \sum_{n=1}^N \log p(\mathbf{y}_n \mid \mathbf{x}_n, \theta) \\ &= \underset{\theta}{\operatorname{argmin}} NLL(\theta) \quad \text{con} \quad NLL(\theta) = -LL(\theta) = -\sum_{n=1}^N \log p(\mathbf{y}_n \mid \mathbf{x}_n, \theta)\end{aligned}$$

1.1.2 MLE con la conjunta

MLE con la conjunta: de un vector de parámetros θ con respecto a un conjunto de N datos $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}$ independientes e idénticamente distribuidos según una fdp (o fp) $p(\mathbf{x}, \mathbf{y} \mid \theta)$

$$\hat{\theta}_{\text{mle}} = \underset{\theta}{\operatorname{argmin}} -\sum_{n=1}^N \log p(\mathbf{x}_n, \mathbf{y}_n \mid \theta)$$

El MLE con la conjunta es distinto del MLE (con la condicional):

$$\hat{\theta}_{\text{mle}} = \underset{\theta}{\operatorname{argmin}} -\sum_{n=1}^N \log p(\mathbf{y}_n \mid \mathbf{x}_n, \theta)$$

$$\begin{aligned}
&= \operatorname{argmin}_{\boldsymbol{\theta}} - \sum_{n=1}^N \log \frac{p(\mathbf{x}_n, \mathbf{y}_n \mid \boldsymbol{\theta})}{p(\mathbf{x}_n \mid \boldsymbol{\theta})} \\
&= \operatorname{argmin}_{\boldsymbol{\theta}} - \sum_{n=1}^N \log p(\mathbf{x}_n, \mathbf{y}_n \mid \boldsymbol{\theta}) - \underbrace{\log p(\mathbf{x}_n \mid \boldsymbol{\theta})}_{\text{depende de toda } \mathbf{y}}
\end{aligned}$$

Cálculo del MLE con la conjunta para clasificación en clases de parámetros independientes: $1 + C$ problemas independientes

Si $y \in \{1, \dots, C\}$ y $\boldsymbol{\theta} = (\{\pi_c\}, \{\boldsymbol{\theta}_c\})$ con $p(y = c \mid \boldsymbol{\theta}) = \pi_c$ y $p(\mathbf{x} \mid y = c, \boldsymbol{\theta}) = p(\mathbf{x} \mid \boldsymbol{\theta}_c)$, entonces:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{mle}} &= \operatorname{argmin}_{\boldsymbol{\theta}} - \sum_{n=1}^N \log p(\mathbf{x}_n, y_n \mid \boldsymbol{\theta}) \\
&= \operatorname{argmin}_{\boldsymbol{\theta}} - \sum_{n=1}^N \log p(y_n \mid \{\pi_c\}) - \sum_{c=1}^C \sum_{n: y_n=c} \log p(\mathbf{x}_n \mid \boldsymbol{\theta}_c) = \{\hat{\pi}_c\}, \{\hat{\boldsymbol{\theta}}_c\}
\end{aligned}$$

$$\begin{aligned}
\{\hat{\pi}_c\} &= \operatorname{argmin}_{\{\pi_c\}} - \sum_{n=1}^N \log p(y_n \mid \{\pi_c\}) \rightarrow \hat{\pi}_c = \frac{N_c}{N} \\
\hat{\boldsymbol{\theta}}_c &= \operatorname{argmin}_{\boldsymbol{\theta}_c} - \sum_{n: y_n=c} \log p(\mathbf{x}_n \mid \boldsymbol{\theta}_c) \quad (c = 1, \dots, C)
\end{aligned}$$

1.2 MLE para la Bernoulli (incondicional)

Datos: $\mathcal{D} = \{y_1, \dots, y_N\}$, $p(y_n \mid \theta) = \text{Ber}(y \mid \theta)$

NLL: de θ con respecto a \mathcal{D}

$$\begin{aligned}
\text{NLL}(\theta) &= -\log \prod_{n=1}^N p(y_n \mid \theta) = -\log \prod_{n=1}^N \theta^{\mathbb{I}(y_n=1)} (1-\theta)^{\mathbb{I}(y_n=0)} \\
&= -\sum_{n=1}^N \mathbb{I}(y_n = 1) \log \theta + \mathbb{I}(y_n = 0) \log(1-\theta) \\
&= -[N_1 \log \theta + N_0 \log(1-\theta)] \quad \text{con} \quad N_1 = \sum_n \mathbb{I}(y_n = 1) \quad \text{y} \quad N_0 = \sum_n \mathbb{I}(y_n = 0)
\end{aligned}$$

Estadísticos suficientes (de los datos): N_1 y N_0 resumen todo lo que necesitamos saber sobre \mathcal{D}

Derivada de la NLL:

$$\frac{d}{d\theta} \text{NLL}(\theta) = \frac{-N_1}{\theta} + \frac{N_0}{1-\theta}$$

MLE: se obtiene igualando la derivada a cero

$$\hat{\theta} = \frac{N_1}{N}$$

Ejemplo: MLE de θ con $N = 100$ datos iid según una $\text{Ber}(y \mid \theta = 0.20)$

In [1]:

```
from scipy.stats import bernoulli
t, N = 0.20, 100; Y = bernoulli(t).rvs(N, random_state=23); print(f'Y = {Y}')
ht = Y.mean(); print(f'ht = {ht:.2f}')
```

```
Y = [0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1
     0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0
     0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
ht = 0.22
```

1.3 MLE para la categórica (incondicional)

Datos: $\mathcal{D} = \{y_1, \dots, y_N\}$, $p(y_n \mid \theta) = \text{Cat}(y \mid \theta)$

NLL: de θ con respecto a \mathcal{D}

$$\begin{aligned} \text{NLL}(\theta) &= -\log \prod_{n=1}^N p(y_n \mid \theta) = -\log \prod_{n=1}^N \prod_{c=1}^C \theta_c^{\mathbb{I}(y_n=c)} \\ &= -\sum_{n=1}^N \sum_{c=1}^C \mathbb{I}(y_n = c) \log \theta_c = -\sum_{c=1}^C N_c \log \theta_c \quad \text{con} \quad N_c = \sum_n \mathbb{I}(y_n = c) \end{aligned}$$

Estadísticos suficientes (de los datos): $\{N_c\}$

Lagrangiana: NLL y la restricción " $1 - \sum_c \theta_c = 0$ " añadida junto con un multiplicador de Lagrange λ

$$\mathcal{L}(\theta, \lambda) = -\sum_c N_c \log \theta_c - \lambda \left(1 - \sum_c \theta_c \right)$$

Gradiente de la Lagrangiana:

$$\frac{\partial \mathcal{L}}{\partial \theta_c} = -\frac{N_c}{\theta_c} \quad y \quad \frac{\partial \mathcal{L}}{\partial \lambda} = -1 + \sum_c \theta_c$$

MLE: se obtiene igualando el gradiente de la Lagrangiana a cero

$$\hat{\theta}_c = \frac{N_c}{N}$$

Ejemplo: MLE de θ con $N = 100$ datos iid según una $\text{Cat}(y \mid \theta)$, $\theta = (0.3, 0.2, 0.5)^t$

```
In [1]: from scipy.stats import multinomial
t, N = [0.3, 0.2, 0.5], 100; Y = multinomial(N, t).rvs(N, random_state=23)
ht = Y.mean(axis=0) / N; print(f'Y[:5,:] = {Y[:5,:]} \n ht = {ht}')
```

```
Y[:5,:] = [[30 26 44]
 [33 17 50]
 [26 23 51]
 [26 20 54]
 [31 19 50]]
ht = [0.2983 0.1992 0.5025]
```

1.4 MLE para la Gaussiana univariada (incondicional)

Datos: $\mathcal{D} = \{y_1, \dots, y_N\}$, $p(y_n \mid \theta) = \mathcal{N}(y \mid \mu, \sigma^2)$

NLL: de $\theta = (\mu, \sigma^2)^t$ con respecto a \mathcal{D}

$$\begin{aligned} \text{NLL}(\mu, \sigma^2) &= -\sum_n \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp \left(-\frac{1}{2\sigma^2} (y_n - \mu)^2 \right) \right] \\ &= \frac{N}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_n (y_n - \mu)^2 \end{aligned}$$

MLE: se obtiene igualando el gradiente de la NLL a cero

$$\begin{aligned} \frac{\partial \text{NLL}}{\partial \mu} &= \frac{1}{\sigma^2} \sum_n (\mu - y_n) = \frac{N}{\sigma^2} (\mu - \bar{y}) \Big|_{\hat{\theta}} = 0 \rightarrow \hat{\mu} = \bar{y} \\ \frac{\partial \text{NLL}}{\partial \sigma} &= \frac{N}{\sigma} - \frac{1}{\sigma^3} \sum_n (y_n - \mu)^2 \Big|_{\hat{\theta}} = 0 \rightarrow \hat{\sigma}^2 = \frac{1}{N} \sum_n (y_n - \hat{\mu})^2 \end{aligned}$$

Estadísticos suficientes: la media empírica \bar{y} y la media de cuadrados empírica, s^2 , pues $\hat{\sigma}^2 = s^2 - \bar{y}^2$

Ejemplo: MLE de θ con $N = 100$ datos iid según una $\mathcal{N}(y \mid \mu, \sigma^2)$, $\mu = 0, \sigma^2 = 1$

In [1]:

```
import numpy as np; from scipy.stats import norm
m, v, N = 0.0, 1.0, 100; Y = norm(m, v).rvs(N, random_state=23)
hm = Y.mean() / N; s2 = np.square(Y).mean(); hv = s2 - hm * hm
print(f'm = {m:.4f} v = {v:.4f} hm = {hm:.4f} hv = {hv:.4f}')
```

m = 0.0000 v = 1.0000 hm = 0.0011 hv = 0.9063

1.5 MLE para la Gaussiana multivariada (incondicional)

Datos: $\mathcal{D} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$, $p(\mathbf{y}_n \mid \theta) = \mathcal{N}(\mathbf{y} \mid \mu, \Sigma)$

NLL: de $\theta = (\text{vec}(\mu); \text{vec}(\Sigma))$ con respecto a \mathcal{D}

$$\begin{aligned} \text{NLL}(\theta) &= - \sum_n \log \left[(2\pi)^{-D/2} |\Sigma|^{-1/2} \exp \left(-\frac{1}{2} (\mathbf{y}_n - \mu)^t \Sigma^{-1} (\mathbf{y}_n - \mu) \right) \right] \\ &= \underbrace{-\frac{ND}{2} \log(2\pi)}_{\text{constante irrelevante}} + \frac{N}{2} \log |\Sigma| + \frac{1}{2} \sum_n (\mathbf{y}_n - \mu)^t \Sigma^{-1} (\mathbf{y}_n - \mu) \end{aligned}$$

MLE de μ : ver [Scalar-by-vector identities \(en formato numerador\)](#)

$$\begin{aligned} \frac{\partial \text{NLL}}{\partial \mu^t} &= \frac{1}{2} \sum_n \frac{\partial (\mathbf{y}_n - \mu)^t \Sigma^{-1} (\mathbf{y}_n - \mu)}{\partial \mu^t} \\ &= \frac{1}{2} \sum_n (\mathbf{y}_n - \mu)^t \Sigma^{-1} \frac{\partial (\mathbf{y}_n - \mu)}{\partial \mu^t} + (\mathbf{y}_n - \mu)^t \Sigma^{-1} \frac{\partial (\mathbf{y}_n - \mu)}{\partial \mu^t} \\ &= \frac{1}{2} \sum_n -2 \Sigma^{-1} (\mathbf{y}_n - \mu) \\ &= -\Sigma^{-1} \sum_n (\mathbf{y}_n - \mu) \Big|_{\hat{\theta}} = \mathbf{0} \rightarrow \hat{\mu} = \bar{\mathbf{y}} \end{aligned}$$

MLE de Σ : ver [Scalar-by-matrix identities \(en formato numerador\)](#)

$$\begin{aligned} \frac{\partial \text{NLL}}{\partial \Sigma} &= \frac{N}{2} \Sigma^{-1} + \frac{1}{2} \sum_n \frac{\partial (\mathbf{y}_n - \mu)^t \Sigma^{-1} (\mathbf{y}_n - \mu)}{\partial \Sigma} \\ &= \frac{N}{2} \Sigma^{-1} + \frac{1}{2} \sum_n \frac{\partial \text{tr}(\Sigma^{-1} (\mathbf{y}_n - \mu)(\mathbf{y}_n - \mu)^t)}{\partial \Sigma} \end{aligned}$$

$$\begin{aligned}
&= \frac{N}{2} \boldsymbol{\Sigma}^{-1} + \frac{1}{2} \sum_n -\boldsymbol{\Sigma}^{-1} (\mathbf{y}_n - \boldsymbol{\mu})(\mathbf{y}_n - \boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1} \Big|_{\hat{\boldsymbol{\theta}}} = \mathbf{0} \\
&\rightarrow N \hat{\boldsymbol{\Sigma}}^{-1} = \hat{\boldsymbol{\Sigma}}^{-1} \sum_n (\mathbf{y}_n - \hat{\boldsymbol{\mu}})(\mathbf{y}_n - \hat{\boldsymbol{\mu}})^t \hat{\boldsymbol{\Sigma}}^{-1} \\
&\rightarrow \hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_n (\mathbf{y}_n - \hat{\boldsymbol{\mu}})(\mathbf{y}_n - \hat{\boldsymbol{\mu}})^t
\end{aligned}$$

Estadísticos suficientes: \bar{y} y $\sum_n \mathbf{y}_n \mathbf{y}_n^t$ pues $\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_n \mathbf{y}_n \mathbf{y}_n^t - \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^t$

Ejemplo: MLE de $\boldsymbol{\theta}$ con $N = 10000$ datos iid según una $\mathcal{N}(\mathbf{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\mu} = \mathbf{0}$, $\boldsymbol{\Sigma} = [2, 1.8; 1.8, 2]$

In [1]:

```
import numpy as np; from scipy.stats import multivariate_normal
m = np.array([.0, .0]); S = np.array([[2., 1.8], [1.8, 2.]]); N = 10000
Y = multivariate_normal(mean=m, cov=S).rvs(N, random_state=23)
hm = Y.mean(axis=0); hS = np.dot(Y.T, Y) / N - np.dot(hm.T, hm)
print(f'hm = {hm}\nhS = {hS}\nnp.cov = {np.cov(Y, rowvar=False, bias=True)}')
```

```
hm = [0.03462052 0.0341259 ]
hS = [[1.97661038 1.77726855]
      [1.77726855 1.98550169]]
np.cov = [[1.97777495 1.77845025]
          [1.77845025 1.98670027]]
```

1.6 MLE para regresión (homocedástica) lineal (condicional)

Datos: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}$, $p(y_n \mid \mathbf{x}_n, \boldsymbol{\theta}) = \mathcal{N}(y_n \mid \mathbf{w}^t \mathbf{x}_n, \sigma^2)$, σ^2 fija

NLL: de \mathbf{w} con respecto a \mathcal{D}

$$\begin{aligned}
\text{NLL}(\mathbf{w}) &= - \sum_n \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp \left(-\frac{1}{2\sigma^2} (y_n - \mathbf{w}^t \mathbf{x}_n)^2 \right) \right] \\
&= \frac{N}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_n (y_n - \mathbf{w}^t \mathbf{x}_n)^2
\end{aligned}$$

Residual sum of squares (RSS): equivalente a la NLL

$$\text{RSS}(\mathbf{w}) = \sum_n r_n^2 \quad \text{con} \quad r_n = y_n - \mathbf{w}^t \mathbf{x}_n$$

Mean squared error (MSE) y root MSE (RMSE): equivalentes a la NLL y RSS

$$\text{MSE}(\mathbf{w}) = \frac{1}{N} \text{RSS}(\mathbf{w}) \quad \text{y} \quad \text{RMSE}(\mathbf{w}) = \sqrt{\text{MSE}(\mathbf{w})}$$

MLE de \mathbf{w} : con notación matricial, $\mathbf{X} = [\mathbf{x}_1^t; \dots; \mathbf{x}_N^t]$ y $\mathbf{y} = (y_1, \dots, y_N)^t$

$$\begin{aligned} \frac{\partial \text{RSS}}{\partial \mathbf{w}^t} &= \frac{\partial (\mathbf{y} - \mathbf{X}\mathbf{w})^t (\mathbf{y} - \mathbf{X}\mathbf{w})}{\partial \mathbf{w}^t} \\ &= \frac{\partial (\mathbf{y}^t - \mathbf{w}^t \mathbf{X}^t) (\mathbf{y} - \mathbf{X}\mathbf{w})}{\partial \mathbf{w}^t} \\ &= \frac{\partial (\mathbf{y}^t \mathbf{y} - 2\mathbf{w}^t \mathbf{X}^t \mathbf{y} + \mathbf{w}^t \mathbf{X}^t \mathbf{X} \mathbf{w})}{\partial \mathbf{w}^t} \\ &= -2\mathbf{X}^t \mathbf{y} + 2\mathbf{X}^t \mathbf{X} \mathbf{w}|_{\hat{\mathbf{w}}} = \mathbf{0} \\ &\rightarrow \mathbf{X}^t \mathbf{X} \hat{\mathbf{w}} = \mathbf{X}^t \mathbf{y} \rightarrow \hat{\mathbf{w}} = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \mathbf{y} \end{aligned}$$

Ordinary least squares (OLS): nombre usual que se le da al MLE de \mathbf{w}

Ejemplo: $p(y_n | x_n, m, c, \sigma^2) = \mathcal{N}(y_n | mx_n + c, \sigma^2)$, $\mathcal{D} = \{(x_n, y_n)\} = \{(0, 0), (1, 0.5), (1, 1.5), (2, 2)\}$

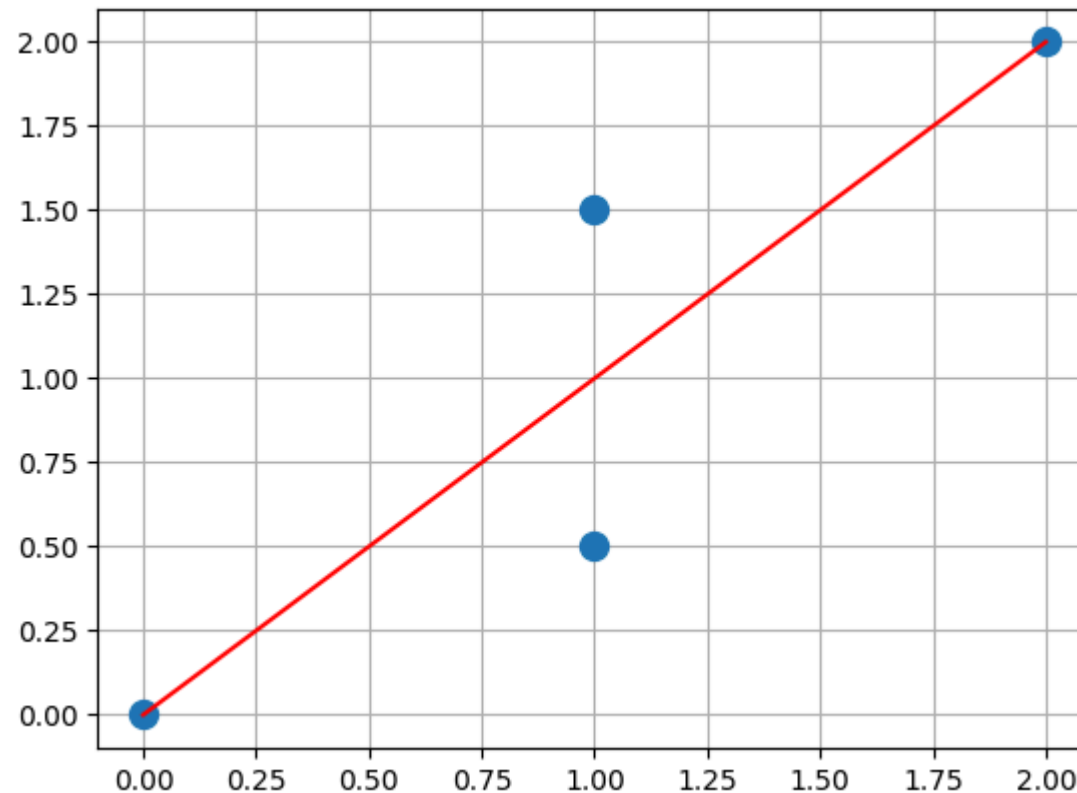
$$\begin{aligned} \mathbf{w} &= (c, m)^t \quad \mathbf{X} = [1, 0; 1, 1; 1, 1; 1, 2] \quad \mathbf{y} = (0, 1/2, 3/2, 2) \\ \mathbf{X}^t \mathbf{X} &= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 4 \\ 4 & 6 \end{pmatrix} \\ (\mathbf{X}^t \mathbf{X})^{-1} &= \frac{1}{8} \begin{pmatrix} 4 & 4 \\ 4 & 6 \end{pmatrix} = \begin{pmatrix} 3/4 & -1/2 \\ 1/2 & 1/2 \end{pmatrix} \\ (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t &= \begin{pmatrix} 3/4 & -1/2 \\ 1/2 & 1/2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 3/4 & 1/4 & 1/4 & -1/4 \\ -1/2 & 0 & 0 & -1/2 \end{pmatrix} \\ \hat{\mathbf{w}} &= \begin{pmatrix} 3/4 & 1/4 & 1/4 & -1/4 \\ -1/2 & 0 & 0 & -1/2 \end{pmatrix} \begin{pmatrix} 0 \\ 1/2 \\ 3/2 \\ 2 \end{pmatrix} = (0, 1)^t \\ \hat{\mathbf{y}} = \mathbf{X} \hat{\mathbf{w}} &= \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \end{pmatrix} \\ \text{RSS}(\hat{\mathbf{w}}) &= (\mathbf{y} - \hat{\mathbf{y}})^t (\mathbf{y} - \hat{\mathbf{y}}) = (0, 1/2, -1/2, 0)^t (0, 1/2, -1/2, 0)^t = 1/2 \end{aligned}$$

$$RSS(w) = (y - \hat{y})(y - \hat{y}) = (0, 1/2, -1/2, 0)(0, 1/2, -1/2, 0)^T = 1/2$$

```
In [1]: import numpy as np; import matplotlib.pyplot as plt
X = np.array([0.0, 1.0, 1.0, 2.0]); N = len(X); Xh = np.c_[np.ones(N), X]
y = np.array([0.0, 0.5, 1.5, 2.0]); hw = np.linalg.inv(Xh.T @ Xh) @ Xh.T @ y
y_pred = Xh @ hw; RSS = np.square(y - y_pred).sum(); print(f'hw = {hw} RSS = {RSS}')
[m, c], [r], _, _ = np.linalg.lstsq(np.c_[X, np.ones(N)], y, rcond=None)
print(f'np.linalg.lstsq: hw = {hw} RSS = {RSS}')
plt.plot(X, y, 'o', markersize=10); plt.plot(X, m*X + c, 'r'); plt.grid(True)
```

hw = [0. 1.] RSS = 0.5

np.linalg.lstsq: hw = [0. 1.] RSS = 0.5



2 Minimización del riesgo empírico

2.1 Definición y ejemplo: minimizar el error de clasificación

2.1.1 Definición

2.1.1 Definición

MLE: minimiza la neg-log-verosimilitud

$$\hat{\theta}_{\text{mle}} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta) \quad \text{con} \quad \mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^N -\log p(\mathbf{y}_n \mid \mathbf{x}_n, \theta)$$

Empirical risk minimization (ERM): generaliza MLE sustituyendo la log-pérdida por una pérdida genérica

$$\hat{\theta}_{\text{erm}} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta) \quad \text{con} \quad \mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^N \ell(\mathbf{y}_n, \theta; \mathbf{x}_n)$$

2.1.2 Ejemplo: minimizar el error de clasificación

Pérdida 01 de un clasificador $f(\mathbf{x}_n; \theta)$:

$$\ell_{01}(\mathbf{y}_n, \theta; \mathbf{x}_n) = \begin{cases} 0 & \text{si } \mathbf{y}_n = f(\mathbf{x}_n; \theta) \\ 1 & \text{si } \mathbf{y}_n \neq f(\mathbf{x}_n; \theta) \end{cases}$$

Riesgo empírico con pérdida 01: es el **error de clasificación** (en entrenamiento)

$$\mathcal{L}_{01}(\theta) = \frac{1}{N} \sum_{n=1}^N \ell_{01}(\mathbf{y}_n, \theta; \mathbf{x}_n)$$

Caso binario: si las clases son $\{-1, +1\}$ y la predicción $\hat{y}_n = f(\mathbf{x}_n; \theta)$

$$\mathcal{L}_{01}(\theta) = \frac{1}{N} \sum_{n=1}^N \ell_{01}(\mathbf{y}_n, \hat{y}_n) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(\mathbf{y}_n \hat{y}_n < 0)$$

2.2 Pérdida subrogada

Surrogate loss function: cota superior de la pérdida 01 que sea ajustada y más fácil de optimizar

2.2.1 Caso binario

Clasificador binario: de log-odds $\eta = f(\mathbf{x}; \theta)$ para etiquetas $\tilde{y} \in \{-1, 1\}$

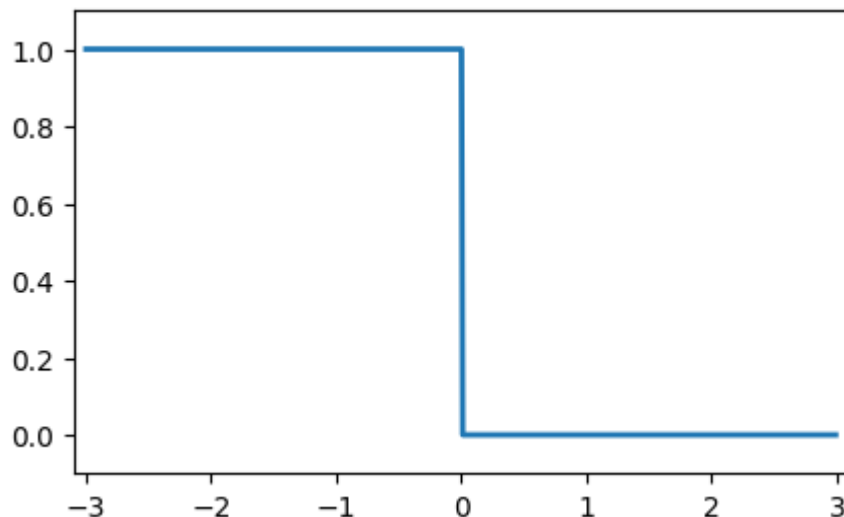
$$p(\tilde{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \sigma(\tilde{y}\eta) = \frac{1}{1 + e^{-\tilde{y}\eta}}$$

Pérdida 01 en el caso binario: $\ell_{01}(\tilde{y}, \eta) = \mathbb{I}(\tilde{y}\eta < 0)$

Margen de seguridad $\tilde{y}\eta$: positivo para que no haya error de clasificación; cuanto mayor sea, mayor seguridad del clasificador

In [1]:

```
import numpy as np; import matplotlib.pyplot as plt
x = np.arange(-3, 3, .01); fig = plt.figure(figsize=(5, 3))
plt.axis([-3.1, 3.1, -0.1, 1.1]); plt.plot(x, x <= 0, linewidth=2);
```



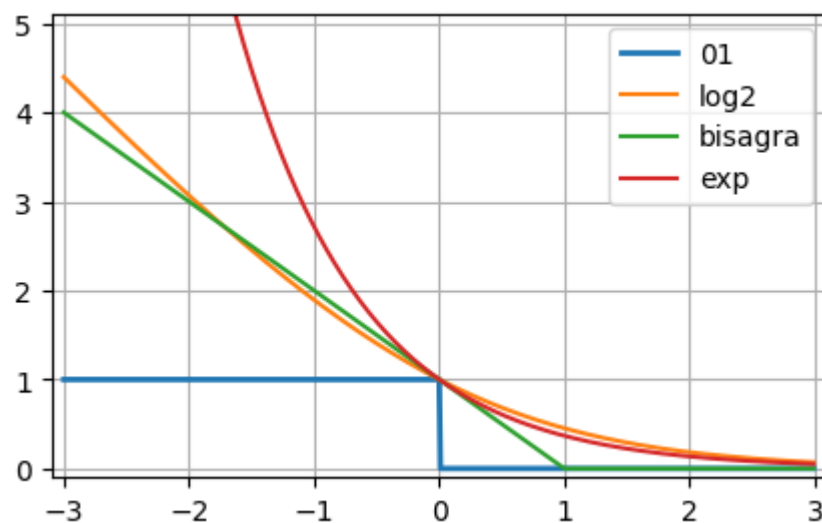
Inconvenientes de la pérdida 01: discontinua en 0 e ignora el margen de seguridad

Pérdidas subrogadas: continuas y tienen en cuenta el margen

Log-pérdida:	$\ell_u(\tilde{y}, \eta) = -\log_2 p(\tilde{y} \mid \eta) = \log_2(1 + e^{-\tilde{y}\eta})$
Hinge (bisagra):	$\ell_{\text{hinge}}(\tilde{y}, \eta) = \max(0, 1 - \tilde{y}\eta) = (1 - \tilde{y}\eta)_+$
Exponencial:	$\ell_{\text{exp}}(\tilde{y}, \eta) = e^{-\tilde{y}\eta}$

In [2]:

```
import numpy as np; import matplotlib.pyplot as plt
x = np.arange(-3, 3, .01); fig = plt.figure(figsize=(5, 3))
plt.axis([-3.1, 3.1, -0.1, 5.1]); plt.grid()
plt.plot(x, x <= 0, linewidth=2, label='01')
plt.plot(x, np.log2(1 + np.exp(-x)), label='log2')
plt.plot(x, (1-x).clip(min=0), label='bisagra')
plt.plot(x, np.exp(-x), label='exp'); plt.legend();
```



3 Regularización

3.1 Definición

3.1.1 Regularización

Sobre-entrenamiento (overfitting): problema fundamental de MLE y, en general, ERM, asociado a la minimización de la pérdida sobre los datos de entrenamiento, ya que conduce a modelos **sobre-entrenados** que **no generalizan** (bien)

Ejemplo de overfitting: probabilidad de obtener cara al lanzar una moneda

- Lanzamos la moneda $N = 3$ veces y obtenemos 3 caras
- El MLE es $\hat{\theta}_{\text{mle}} = N_1 / (N_0 + N_1) = 3 / (0 + 3) = 1$
- Si usamos $\text{Ber}(y \mid \hat{\theta}_{\text{mle}})$ para predecir, predeciremos cara en todos los lanzamientos futuros, cosa bastante inverosímil

Muchos parámetros suele conducir a overfitting: ya que el modelo tiene suficientes parámetros para explicar los datos, por lo que acaba pareciéndose mucho a la empírica y no es capaz de generalizar

Regularización: añade una **penalización (penalty)** a la pérdida mediante alguna forma de **penalización de complejidad** $C(\theta)$ cuyo peso depende de un **parámetro de regularización** $\lambda \geq 0$

$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = \left[\frac{1}{N} \sum_n \ell(\mathbf{y}_n, \boldsymbol{\theta}; \mathbf{x}_n) \right] + \lambda C(\boldsymbol{\theta})$$

Penalización de complejidad usual: $C(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta})$ donde $p(\boldsymbol{\theta})$ es un **prior** más o menos plano

3.1.2 Estimación maximum a posteriori (MAP)

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{\text{map}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(\boldsymbol{\theta} \mid \mathcal{D}) && \text{asumimos } p(\boldsymbol{\theta} \mid \mathcal{D}) = \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{\text{map}}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(\mathcal{D} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} -\frac{1}{N} \log p(\mathcal{D} \mid \boldsymbol{\theta}) - \frac{1}{N} \log p(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \text{NLL}(\boldsymbol{\theta}) - \lambda \log p(\boldsymbol{\theta}) && \text{con log-pérdida y } \lambda = \frac{1}{N} \end{aligned}$$

3.2 Elección del regularizador mediante validación

Efecto del parámetro de regularización λ :

- λ **demasiado pequeño:** equivale a minimizar el riesgo empírico y acabar con un modelo **sobre-ajustado**
- λ **demasiado grande:** equivale a no desviarse mucho del prior y acabar con un modelo **sub-ajustado**

Validación: método más popular y sencillo para escoger λ

- Dividimos los datos en un conjunto de **entrenamiento** $\mathcal{D}_{\text{train}}$ y otro de **validación o desarrollo** $\mathcal{D}_{\text{valid}}$
- Por cada valor de interés de λ , el modelo se ajusta con $\mathcal{D}_{\text{train}}$ y se evalúa en validación
- Elegimos el λ que proporciona mejor rendimiento en validación

Validación con más detalle:

- Riesgo empírico regularizado con un conjunto de datos \mathcal{D} : $R_\lambda(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta})) + \lambda C(\boldsymbol{\theta})$
- Por cada valor de interés de λ , obtenemos el estimador: $\hat{\boldsymbol{\theta}}_\lambda(\mathcal{D}_{\text{train}}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} R_\lambda(\boldsymbol{\theta}, \mathcal{D}_{\text{train}})$
- **Riesgo poblacional estimado como riesgo en validación:** $R_\lambda^{\text{val}} = R_0(\hat{\boldsymbol{\theta}}_\lambda(\mathcal{D}_{\text{train}}), \mathcal{D}_{\text{valid}})$
- Escogemos el regularizador óptimo en validación: $\lambda^* = \underset{\lambda \in \mathcal{S}}{\operatorname{argmin}} R_\lambda^{\text{val}}$
- Finalmente reajustamos el modelo con todo $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{valid}}$: $\hat{\boldsymbol{\theta}}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} R_{\lambda^*}(\boldsymbol{\theta}, \mathcal{D})$

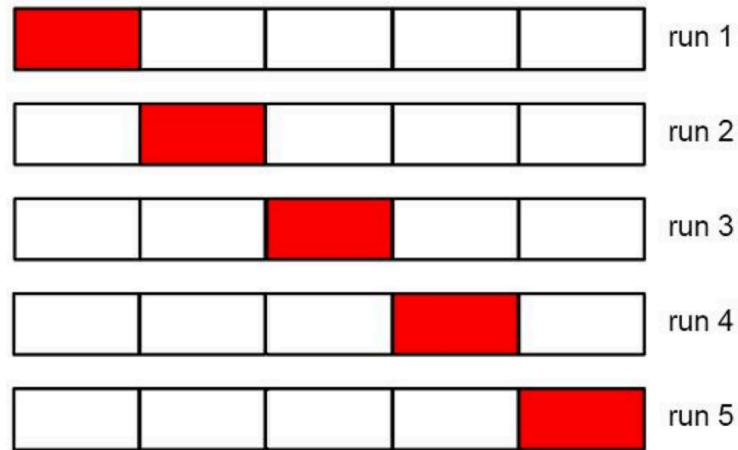
3.3 Validación cruzada

3.3 validación cruzada

Cross validation (CV): divide los datos en K bloques y, por cada bloque $k \in \{1, \dots, K\}$, ajusta el modelo con todos los bloques menos el k -ésimo, el cual se usa como conjunto de validación

$$R_{\lambda}^{\text{CV}} = \frac{1}{K} \sum_k R_0(\hat{\theta}_{\lambda}(\mathcal{D}_{-k}), \mathcal{D}_k)$$

Ejemplo: validación cruzada con $K = 5$ bloques



Leave-one-out (LOO) cross validation: CV con $K = N$

Uso de CV: hallamos un λ óptimo, $\hat{\lambda}$, y luego re-estimamos θ con todos los datos y $\hat{\lambda}$

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmin}} R_{\lambda}^{\text{CV}} \quad \text{y, luego,} \quad \hat{\theta} = \underset{\theta}{\operatorname{argmin}} R_{\hat{\lambda}}(\theta, \mathcal{D})$$

3.3.1 La regla "un error estándar"

Propósito: añadir una medida de incertidumbre al estimador CV del riesgo

Error estándar de la media: a partir del estimador CV del riesgo de cada muestra n , $L_n = \ell(\mathbf{y}_n, f(\mathbf{x}_n; \hat{\theta}_{\lambda}(\mathcal{D}_{-n})))$

$$\text{se}(\hat{\mu}) = \frac{\hat{\sigma}}{\sqrt{N}} \quad \text{donde} \quad \hat{\mu} = \frac{1}{N} \sum_n L_n \quad \text{y} \quad \hat{\sigma}^2 = \frac{1}{N} \sum_n (L_n - \hat{\mu})^2$$

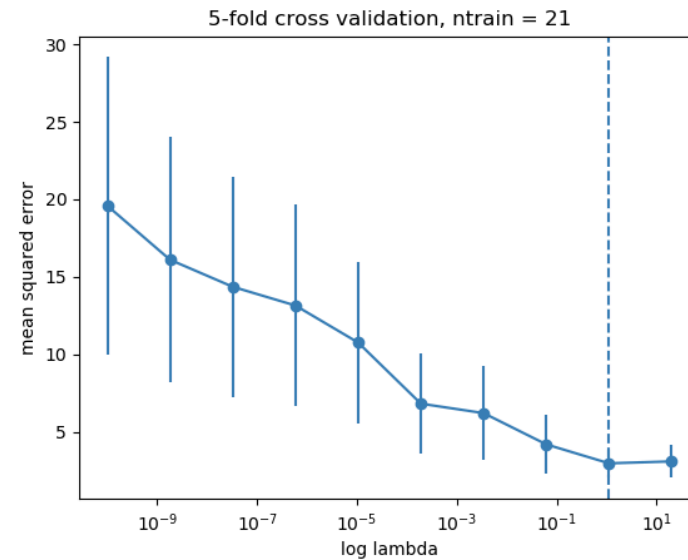
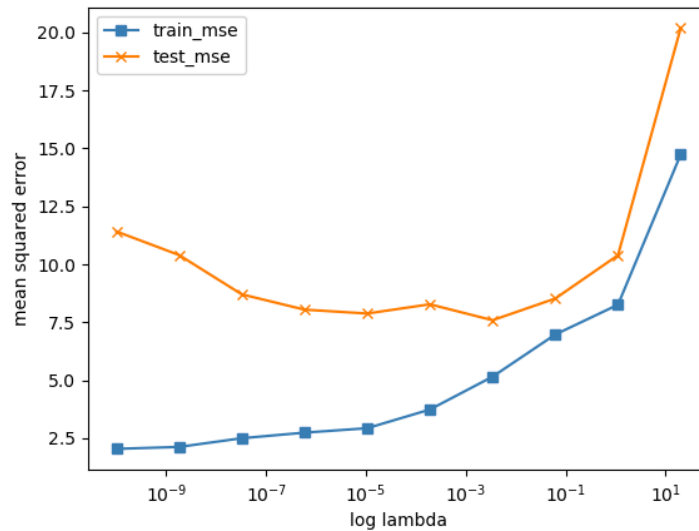
Una standard error rule es emplear para la selección de un modelo entre varios candidatos

One-standard error rule: se emplea para la selección de un modelo entre varios candidatos

- Primero se aplica CV a cada modelo, obteniéndose la media y error estándar (de la media) a partir de sus riesgos estimados
- Se selecciona el modelo más simple cuyo riesgo no sea mayor que un error estándar por encima del riesgo del mejor modelo

3.3.2 Ejemplo: regresión de cresta

- Izquierda: el MSE de test tiene forma de U; primero decrece al aumentar λ , pero luego crece a causa del subajuste
- Derecha: MSE estimado con 5-bloques CV y error estándar de la media en barras verticales; mínimo cercano al óptimo en test

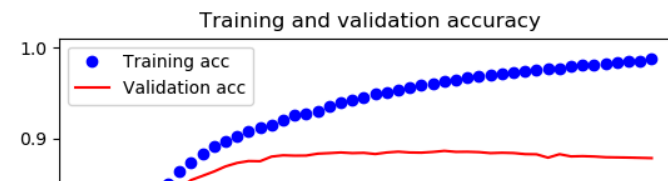
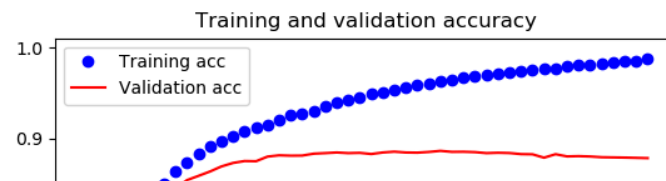


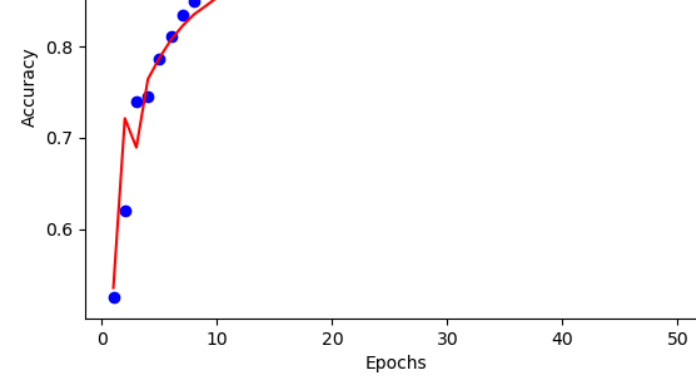
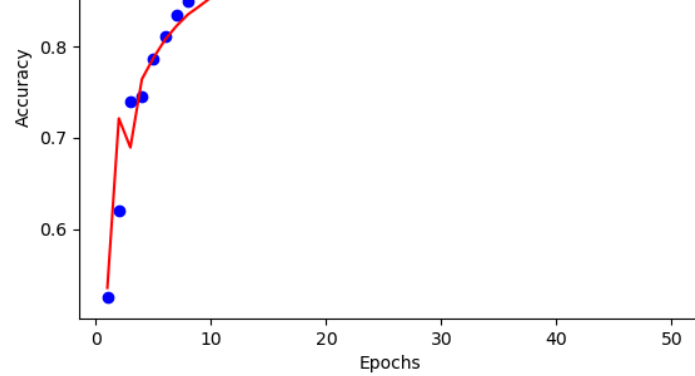
3.4 Terminación temprana

Asunción: el ajuste de parámetros se realiza mediante un algoritmo de optimización iterativa

Early stopping: valida el rendimiento del modelo tras cada iteración y termina antes de convergencia si se observa sobre-entrenamiento

Ejemplo: pérdida y precisión de un clasificador de texto en IMDB





3.5 Uso de más datos

Datos y sobre-ajuste: dado un un modelo de complejidad fija, la posibilidad de sobre-ajuste decrece al aumentar los datos

Ejemplo: MSE en train y test de en función de N , para regresión de cresta con modelos de diferente complejidad

