

Actividad 6

Leonardo Flores Torres

31 de octubre de 2022

1. Ajuste de distribuciones de probabilidad a datos de entrenamiento y la distribución predictiva para una distribución Gaussiana unidimensional. Defina $N > 10$ puntos aleatorios extraídos de una distribución gaussiana $G_x(\mu_0, \sigma_0)$, con $\mu_0 = 5$ y $\sigma_0 = 1.5$, $\text{RandomG}_x(\sigma_0, \mu_0)$.

Solución:

Una distribución normal univariada $\text{Norm}_x[\sigma, \mu]$ está dada por

$$\text{Norm}_x[\sigma, \mu] = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{1}{2} \frac{(x - \mu)^2}{\sigma^2} \right], \quad (1)$$

donde μ es la media y σ es la desviación estándar. Se generó una distribución normal univariada centrada en $\mu_0 = 5$ con desviación estándar $\sigma_0 = 1.5$, como se muestra en la figura 1, con dominio $D_0 : \{x \mid x \in [\mu_0 - 4\sigma_0, \mu_0 + 4\sigma_0]\}$. De esta distribución normal se realizó un muestreo para extraer un número $N = 30$ de datos los cuales son el conjunto de entrenamiento $\{x_i\}_{i=1}^{30}$ usado en el resto de los incisos a responder en esta actividad.

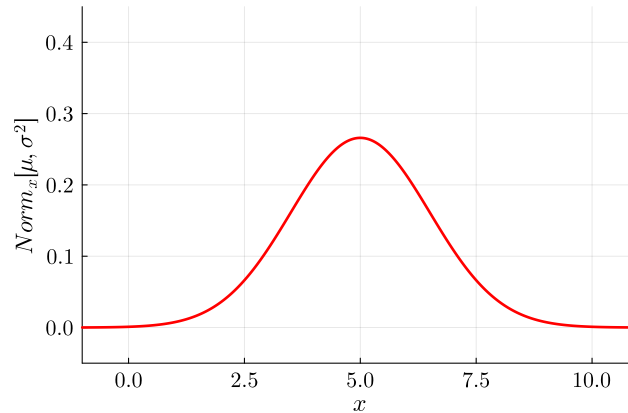


Figura 1: Distribución gaussiana de probabilidad usada para extraer los datos de entrenamiento.

Solamente para clarificar, los parámetros que maximizan la verosimilitud $\{\hat{\mu}, \hat{\sigma}\}$ se encuentran al computar las derivadas e igualando a cero para encontrar un máximo, respecto a cada uno de los parámetros $\{\mu, \sigma\}$, del producto de las verosimilitudes individuales para cada uno de los puntos del conjunto muestral $\{x_i\}_{i=1}^N$. Como se está usando una distribución normal univariada se tiene que

la verosimilitud es igual a

$$\Pr(x_{1...N}|\mu, \sigma^2) = \prod_{i=1}^N \text{Norm}_{x_i} [\mu, \sigma^2] . \quad (2)$$

Después de encontrar las derivadas respecto a cada uno de los parámetros e igualando a cero se obtiene que los parámetros que maximizan la verosimilitud son

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^N x_i , \quad (3)$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{\mu})^2 . \quad (4)$$

Para generar estos datos se definen los parámetros iniciales μ_0 y σ_0 , se define el dominio en x_0 , y el muestreo se computa en `xsample`. También se computan los valores de la gaussiana en el dominio, i.e., $G_{x_0}(\mu_0, \sigma_0)$. Es importante mencionar que el dominio será el mismo en todas las demás distribuciones que se generen, esto quiere decir $D_i = D_0$ para $i = 1, 2, 3$ donde i denota el conjunto de condiciones $\{\mu_i, \sigma_i\}$ mencionadas en el segundo inciso de la actividad.

```

1  julia> using Revise, NormalDist; nd = NormalDist;
2  julia> begin
3      # Parametros iniciales de la gaussiana
4      μ0 = 5
5      σ0 = 1.5
6      # Dominio de la gaussiana original
7      x0 = range(μ0 - 4 * σ0, μ0 + 4 * σ0, 2000)
8      # Evaluacion de la gaussiana en el dominio G(x0)
9      w0 = nd.normalDist(x0, μ0, σ0)
10     # Numero de puntos para muestreo
11     npts = 30
12     # Muestreo de puntos
13     xsample = nd.randomSample(x0, w0, npts)
14     # Grafica de la gaussiana
15     fig = nd.plotNormal(x0, w0)
16     end
    
```

Nótese que en ningún momento se ha mencionado que es necesario restringir el conjunto de entrenamiento a valores únicos, por lo que está permitido tener puntos repetidos dentro del conjunto aleatorio de entrenamiento $\{x_i\}_i^{30}$.

- Grafique 3 figuras como la lámina 16 del Prince, diapositiva 4, donde se muestren los N puntos, y los valores del likelihood para 3 Gaussianas considerando

- $\mu_1 = 3, \sigma_1 = 1,$
- $\mu_2 = 6, \sigma_2 = 1.6,$
- $\mu_3 = 5.1, \sigma_3 = 1.4.$

Grafique las Gaussianas sobre los puntos de entrenamiento y su ordenada para cada Gaussianas, $\text{Gauss}_x(\mu_i, \sigma_i)$.

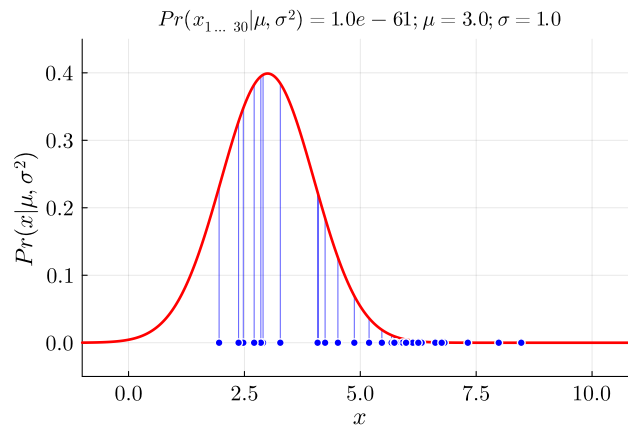
Solución:

Un ejemplo de como se computaron todas las distribuciones se muestra a continuación:

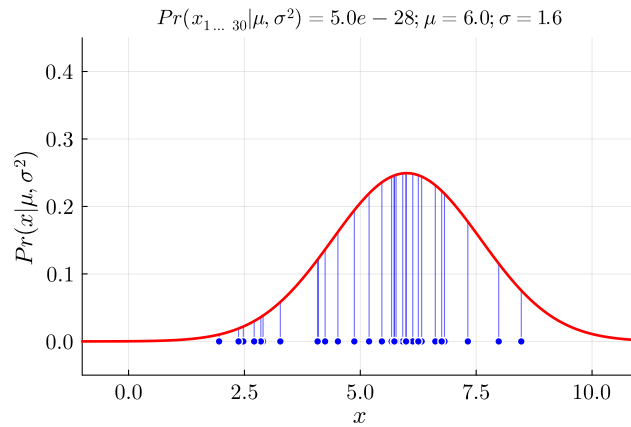
```

17 julia> begin
18     # Parametros para el i-esimo caso
19     μi = 6      # parametro μi
20     σi = 1.6    # parametro σi
21     # Evaluacion de la i-esima gaussiana con nuevos parametros sobre el conjunto de datos
22     ↪ muestreados
23     wsample = nd.normalDist(xsample, μi, σi)
24     # Computo de la i-esima gaussiana con nuevos parametros
25     wi = nd.normalDist(x0, μi, σi)
26     # Likelihood de los datos xsample respecto a los i-esimos parametros
27     lh = nd.likelihood(xsample, μi, σi)
28     # Grafica de la i-esima gaussiana con los datos muestreados
29     fig = nd.plotGuess(xsample, wsample, x0, wi, lh, μi, σi; maxlh = false)
    end
    
```

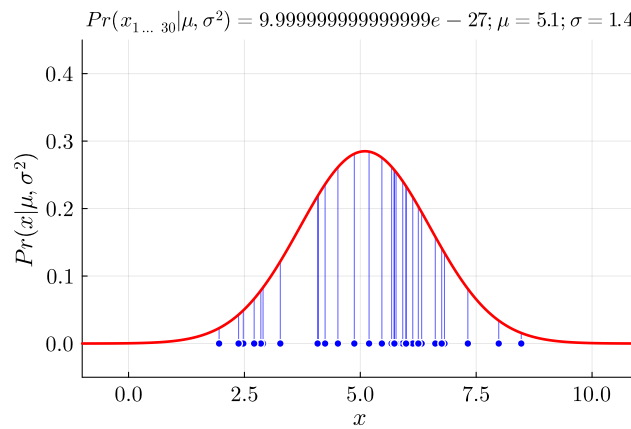
Las gaussianas obtenidas con sus respectivos parámetros se muestran en la figura 2, donde también se incluyó la gráfica para la gaussiana con los parámetros que maximizan la verosimilitud $\{\hat{\mu}, \hat{\sigma}\}$. Además, cada gráfica incluye su verosimilitud dependiendo de sus parámetros $Pr(x_{1...30} | \mu_i, \sigma_i^2)$. Para el caso de $\{\hat{\mu}, \hat{\sigma}\}$ solamente se realizaron las siguientes asignaciones `μi = nd.maxμ(xsample)` y `σi = nd.maxσ(xsample)` en las líneas 19 y 20, y se cambia el parámetro `maxlh = true` de la función donde se genera la gráfica para decir al programa que cambie el título de la gráfica.



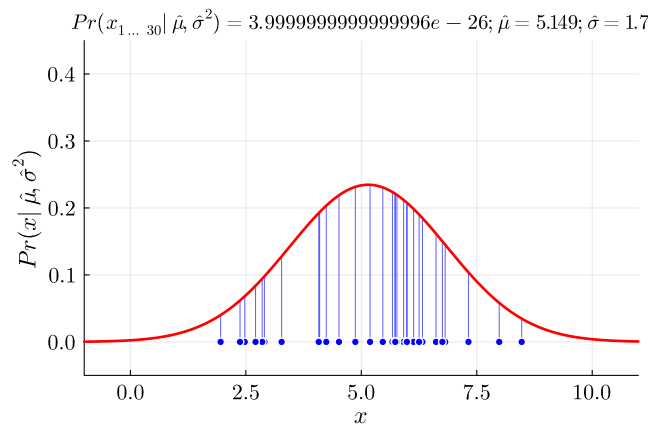
(a) Distribución $G_x(\mu_1, \sigma_1)$ para los parámetros $\mu_1 = 3, \sigma_1 = 1$.



(b) Distribución $G_x(\mu_2, \sigma_2)$ para los parámetros $\mu_2 = 6, \sigma_2 = 1.6$.



(c) Distribución $G_x(\mu_3, \sigma_3)$ para los parámetros $\mu_3 = 5.1, \sigma_3 = 1.4$.



(d) Distribución $G_x(\hat{\mu}, \hat{\sigma})$ para los parámetros que maximizan el likelihood, $\hat{\mu}, \hat{\sigma}$.

Figura 2: Gráficas de las gaussianas requeridas para el segundo punto de la actividad con sus respectivos parámetros.

Nótese que para el conjunto de puntos muestreados $\{x_i\}_{i=1}^{30}$ se obtiene un buen valor para la verosimilitud, con parámetros $\{\hat{\mu} = 5.053, \hat{\sigma} = 1.885\}$ aunque estos valores dependen en gran medida

de los puntos dentro del conjunto muestral.

Por completez se agrega el ejemplo de como computar la distribución con los parámetros mostrados en la figura 2d:

```

30 julia> begin
31     # Parametros para el caso de maximizacion
32     μi = nd.maxμ(xsample)
33     σi = nd.maxσ(xsample)
34     # probabilidades asociadas al conjunto muestral
35     wsample = nd.normalDist(xsample, μi, σi)
36     # Computo de la gaussiana con parametros maximizados
37     wi = nd.normalDist(x0, μi, σi)
38     # Likelihood de los datos xsample respecto a los parametros
39     lh = nd.likelihood(xsample, μi, σi)
40     # Grafica de la gaussiana
41     fig = nd.plotGuess(xsample, wsample, x0, wi, lh, μi, σi; maxlh = true)
42 end

```

3. Genere el mapa térmico de las probabilidades considerando $\mu \in [2.5, 6.5]$, y $\sigma \in [0, 2]$, dividiendo ambos intervalos en 10 partes. Coloque un marcador en el punto de máxima verosimilitud.
4. Calcule, usando el método de Maximum Likelihood, los parámetros $\hat{\mu}$ y $\hat{\sigma}^2$ que tanto coincide con el mapa térmico.

Solución:

Los incisos 3 y 4 se resolvieron de manera conjunta ya que al añadir un marcador en el mapa de calor que indique donde se encuentra el punto de máxima verosimilitud este se tuvo que haber computado de antemano. Me tomé la libertad de cambiar el número de divisiones a 150 en que se subdividen los intervalos para μ y σ . La instrucción escrita en `julia` que genera el mapa de calor se muestra a continuación:

```

43 julia> begin
44     # Numero de particiones para los intervalos de μ y σ
45     partitions = 150
46     # Intervalos de μ y σ
47     μint = (2.5, 6.5)
48     σint = (0, 2)
49     # Deltas de μ y σ
50     dμ = (μint[2] - μint[1]) / partitions
51     dσ = (σint[2] - σint[1]) / partitions
52     # Dominios de μ y σ
53     μrange = range(μint[1] + dμ/2, μint[2] - dμ/2, partitions)
54     σrange = range(σint[1] + dσ/2, σint[2] - dσ/2, partitions)
55     # Mallado para obtener un espacio bidimensional
56     (μmesh, σmesh) = nd.ndgrid(μrange, σrange)
57     # Valores del likelihood en el mallado
58     lhmesh = nd.likelihood.(Ref(xsample), μmesh, σmesh)
59     # Parametros que maximizan el likelihood y el likelihood maximo
60     μmax = nd.maxμ(xsample)
61     σmax = nd.maxσ(xsample)

```

```

62     lhmax = nd.likelihood(xsample, μmax, σmax)
63     # Grafica del mallado
64     fig = nd.plotHeat([3, 6, 5.1], [1, 1.6, 1.4], μmax, σmax, μrange, σrange, lhmesh, npts)
65     end
    
```

El mapa de calor se muestra en la figura 3. El punto de máxima verosimilitud con parámetros $\{\hat{\mu}, \hat{\sigma}\}$ está marcado con una estrella de color cyan, mientras que los demás puntos marcados con círculos color magenta corresponden a las verosimilitudes con sus conjuntos respectivos de parámetros $\{\mu_i, \sigma_i\}$. Los valores para las verosimilitudes de cada uno de los casos se puede observar en el título

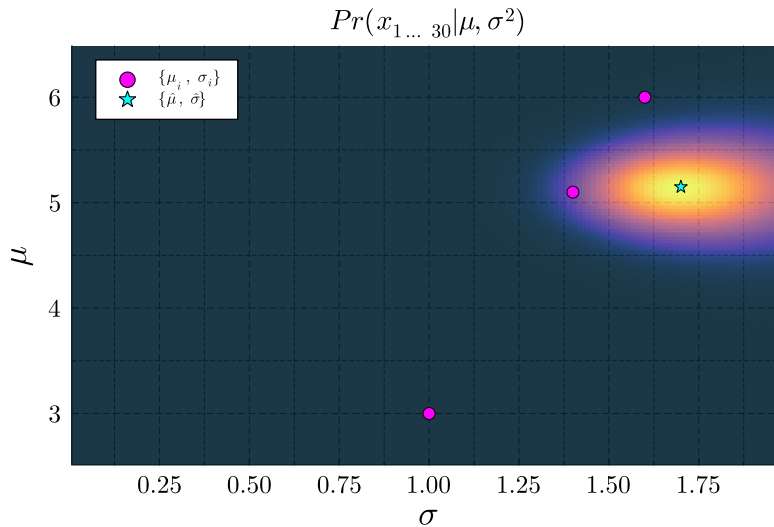


Figura 3: Mapa de calor incluyendo las verosimilitudes de sus respectivos parámetros.

de las figuras 2a a 2d con sus respectivos parámetros. El muestreo no es perfecto, mientras más puntos se obtengan para el muestreo mejor será la aproximación para la máxima verosimilitud. Aún así, al computar las distribuciones gaussianas $G_x(\mu_i, \sigma_i)$ surgía frecuentemente el caso en que la máxima verosimilitud era menor que la verosimilitud para el caso $\{\mu_3 = 5.1, \sigma_3 = 1.4\}$. Atribuyo este comportamiento como resultado del muestreo, pero la máxima verosimilitud siempre parecía encontrarse en el centro del elipsoide del mapa de calor.

Incluso se puede encontrar el valor del máximo likelihood correspondiente al mallado, se usa la función `argmax` sobre la variable donde se guardó el mallado la cual regresa índices y éstos se usan para encontrar los valores correspondientes de μ y σ en ese punto de la malla.

```

66 julia> μmeshi, σmeshi = argmax(lhmesh) ▷ x → (μrange[x[1]], σrange[x[2]])
67 (5.153333333333333, 1.7)
68
69 julia> μmax, σmax
70 (5.149274637318659, 1.700282387563139)
71
72 julia> abs(μmeshi - μmax)
73 0.0040586960146740125
74
75 julia> abs(σmeshi - σmax)
76 0.00028238756313903046
    
```

Se puede observar que la diferencia para μ es de 0.4 por ciento, mientras que la diferencia de σ es del 0.02 por ciento. Para concluir con el trabajo solamente se quisiera recordar al lector que las instrucciones mencionadas sobre como se calcularon los distintos pasos de la actividad se hicieron dentro del REPL de julia, también se incluye el módulo desarrollado para esta actividad en el Apéndice.

Apéndice

```

1  module NormalDist
2
3  using StatsBase
4  using Plots
5  using LaTeXStrings
6  using LazyGrids # To generate lazy grids with low memory allocation
7
8
9  """
10     normalDist(x::Real,  $\mu$ ,  $\sigma$ )
11
12     Computes the corresponding probability of a single point 'x' using a
13     normal distribution with a given mean ' $\mu$ ' and the squared root of the
14     variance ' $\sigma$ '.
15     """
16     function normalDist(x::Real,  $\mu$ ,  $\sigma$ )
17         exparg = - (x -  $\mu$ )^2 / (2 *  $\sigma$ ^2)
18         den = sqrt(2 *  $\pi$  *  $\sigma$ ^2)
19         return exp(exparg) / den
20     end
21
22
23     """
24     normalDist(xpts::Array,  $\mu$ ,  $\sigma$ )
25
26     Computes the corresponding probability from a list of points 'xpts' using a
27     normal distribution with a given mean ' $\mu$ ' and the squared root of the
28     variance ' $\sigma$ '.
29     """
30     function normalDist(xpts::Union{Array, StepRangeLen},  $\mu$ ,  $\sigma$ )
31         return normalDist.(xpts,  $\mu$ ,  $\sigma$ )
32     end
33
34
35     """
36     likelihood(xpts,  $\mu$ ,  $\sigma$ )
37
38     Computes the likelihood of a given array of points 'xpts' with respect of
39     a normal distribution with parameters ' $\mu$ ' and ' $\sigma$ '.
40     """
41     function likelihood(xpts,  $\mu$ ,  $\sigma$ )
42         npts = length(xpts)
43         exparg = - sum((xpts .-  $\mu$ ).^2) / (2 *  $\sigma$ ^2)
44         den = (2 *  $\pi$  *  $\sigma$ ^2)^(npts / 2)

```

```

45     return exp(exparg) / den
46 end
47
48
49 """
50     maxμ(xpts)
51
52 Computes the mean value 'μ' that maximizes the likelihood.
53 """
54 function maxμ(xpts)
55     npts = length(xpts)
56     return sum(xpts) / npts
57 end
58
59
60 """
61     maxσ(xpts)
62
63 Computes the squared-root of the variance 'σ' that maximizes
64 the likelihood.
65 """
66 function maxσ(xpts)
67     npts = length(xpts)
68     return (sum((xpts .- maxμ(xpts)).^2) / npts) > sqrt
69 end
70
71
72 """
73     randomSample(xpts, weights, npts)
74
75 Returns the desired number of points 'npts' from a list of points 'xpts',
76 taking into account a list of their weights 'weights' from any given
77 distribution.
78 """
79 function randomSample(xpts, weights, npts)
80     return sample(xpts, Weights(weights), npts)
81 end
82
83 function plotNormal(x, w)
84     fig = plot(
85         tickfont=(12, "Computer Modern"),
86         xlim = (x[begin], x[end]),
87         ylim = (-0.05, 0.45),
88         dpi = 400,
89         size = (600, 400),
90     )
91
92     plot!(x, w,
93         linecolor = :red,
94         lw = 2.5,
95         label = "",
96         xlabel = L"x",
97         ylabel = L"Norm_{x}[\mu, \sigma^2]",
98         xtickfontsize = 13,

```



```

99     ytickfontsize = 13,
100     xguidefontsize = 15,
101     yguidefontsize = 15,
102     titlefontsize = 13,
103 )
104
105 return fig
106 end
107
108 function plotGuess(xsample, wsample, x, w, lh,  $\mu$ i,  $\sigma$ i; maxlh = false)
109     npts = length(xsample)
110
111     round $\mu$  = round( $\mu$ i, sigdigits=4)
112     round $\sigma$  = round( $\sigma$ i, sigdigits=4)
113     roundlh = round(lh, sigdigits=1)
114
115     # theme(:ggplot2)
116     fig = plot(
117         tickfont=(12, "Computer Modern"),
118         xlim = (x[begin], x[end]),
119         ylim = (-0.05, 0.45),
120         dpi = 400,
121         size = (600, 400),
122     )
123
124     for i in 1:npts
125         plot!(
126             [xsample[i], xsample[i]],
127             [0, wsample[i]],
128             label = "",
129             linecolor = :blue,
130             linealpha = 0.6,
131         )
132     end
133
134     # Just some formatting depending on if the maximum likelihood is
135     # plotted or not
136      $\mu$ str = maxlh == false ? " $\mu$ " : raw"\hat{\mu}"
137      $\sigma$ str = maxlh == false ? " $\sigma$ " : raw"\hat{\sigma}"
138     title = raw"Pr( $x_{1 \dots}$  *  $\$(npts)$  * raw"} | " *  $\mu$ str * raw", " *
139          $\sigma$ str * raw"^2) = " *  $\$(roundlh)$  * raw"; " *  $\mu$ str * " = " *
140          $\$(round\mu)$ ; " *  $\sigma$ str * raw" = " *  $\$(round\sigma)$ "
141     ylabel = raw"Pr( $x$  | " *  $\$(\mu$ str) * ", " *  $\$(\sigma$ str) * raw"^2)"
142     plot!(x, w,
143         linecolor = :red,
144         lw = 2.5,
145         label = "",
146         xlabel = L"x",
147         ylabel = latexstring(ylabel),
148         title = latexstring(title),
149         xtickfontsize = 13,
150         ytickfontsize = 13,
151         xguidefontsize = 15,
152         yguidefontsize = 15,

```

```

153     titlefontsize = 13,
154 )
155
156 scatter!(xsample, zeros(npts),
157     label = "",
158     mc = :blue,      # marker color
159     msc = :white,    # marker stroke color
160     msw = 2,         # marker stroke width
161 )
162
163     return fig
164 end
165
166
167 function plotHeat(μi, σi, μmax, σmax, μrange, σrange, lhmesh, npts)
168     title = raw"Pr(x_{1\ldots} * "$(npts)" * raw"} | μ, σ^2)"
169     fig = plot(
170         xlim = (σrange[begin], σrange[end]),
171         ylim = (μrange[begin], μrange[end]),
172         dpi = 400,
173         size = (600, 400),
174         xlabel = L"\sigma",
175         ylabel = L"\mu",
176         title = latexstring(title),
177         xguidefontsize = 17,
178         yguidefontsize = 17,
179         tickfont = (12, "Computer Modern"),
180         grid = :dash,
181         gridlinewidth = 0.6,
182         gridalpha = 0.95,
183         minorgrid = :dash,
184         minorgridlinewidth = 0.4,
185         minorticks = 2,
186         minorgridalpha = 0.95,
187         legend = :topleft,
188     )
189
190     heatmap!(σrange, μrange, lhmesh,
191         c = cgrad(:thermal, rev = false, alpha=0.9), # colormap
192         colorbar = :none,
193     )
194
195     scatter!(σi, μi,
196         label = L"\mu_{i}\ ,\ \sigma_{i}\}",
197         mc = :magenta,
198         ms = 5,
199         msw = 1,
200         msc = :black,
201     )
202
203     scatter!([σmax], [μmax],
204         label = L"\hat{\mu}\ ,\ \hat{\sigma}\}",
205         mc = :cyan,
206         ms = 5,

```

```
207     shape = :star5,  
208     msw = 1,  
209     msc = :black  
210 )  
211  
212     return fig  
213 end  
214  
215  
216 end # module NormalDist
```

Referencias

- [1] Simon JD Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.
- [2] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 9 2017.