Estadística III para Ingenieros de Sistemas

Jose Daniel Ramirez Soto 2023
jdr2162@columbia.edu

Agenda

- **anuncios varios**
    - Tarea 2 entrega lunes 24 de Abril (Preguntas)
- **modelos de analitica (machine learning-ML) Supervisado**
    - **Regresión logística**
        - **Matemática de la regresión logística**
            - **DeepDive, Gradiente descendiente**
        - **Árboles**
            - **Árboles simples**
            - **GMB**

Logistic Regression, DeepDive gradiente

Utilizando los datos de 2 exámenes para predecir si el estudiante aprueba el curso. Vemos que el log likelihood es mayor si el modelo tiene menos errores en los datos de entrenamiento. Monotonically

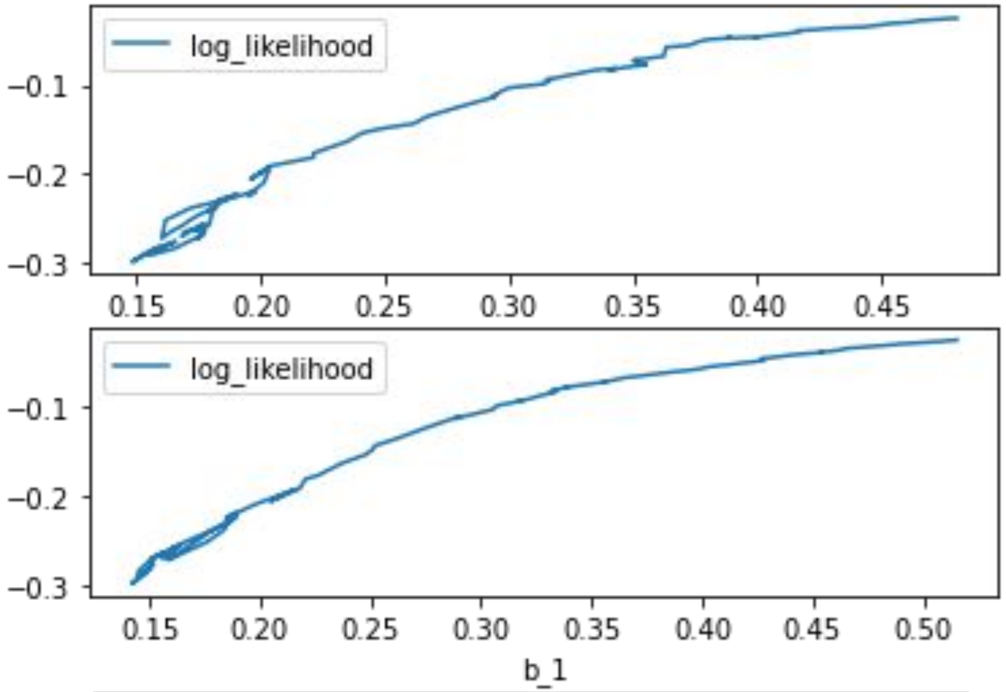
$$y = \frac{1}{(1+e^{-(b1x_1+b2x_2+b0)})}$$

Cada dato : $p(y =" c " |experimentos) = p(y_1 =" C ") * p(y_2 =" C ")$

$$p(y=1|d) = 1/n \prod_{i=1}^{n}(1-p(y))^{1-y}.\,(p(y))^{y}$$

$$l = \prod_{i=1}^{n}(1 - \frac{1}{(1+e^{-(b1x+b2x_2+b_0)})})^{(1-y)} * (\frac{1}{(1+e^{-(b1x+b2x_2+b_0)})})^{y}$$

$$\log(l) = \sum_{i=1}^{n} +(1-y)\log(1 - \frac{1}{(1+e^{-(b_1x+b_2x_2+b_0)})}) + y\log(\frac{1}{(1+e^{-(b_1x+b_2x_2+b_0)})})$$



Bernoulli, mu= p var= p(1-p)

$$\mathrm{P}[X=x] = p^{x}(1-p)^{1-x} \qquad x = 0, 1$$

Logistic Regression, DeepDive gradiente

$$\frac{dl}{db_1} = \frac{dl}{du}\frac{du}{db_1}$$

$$\frac{dl}{db_1} = y\log([1+e^{-(b_1x+b_2x_2+b_0)}]^{-1}) + (1-y)\log(\frac{e^{-(b_1x+b_2x_2+b_0)}}{1+e^{-(b_1x+b_2x_2+b_0)}})$$

$$= -x_1 - (1+e^{-(b_1x_1+b_2x_2+c)})^{-1}(e^{-(b_1x_1+b_2x_2+c)})(-x_1)) + yx_1$$

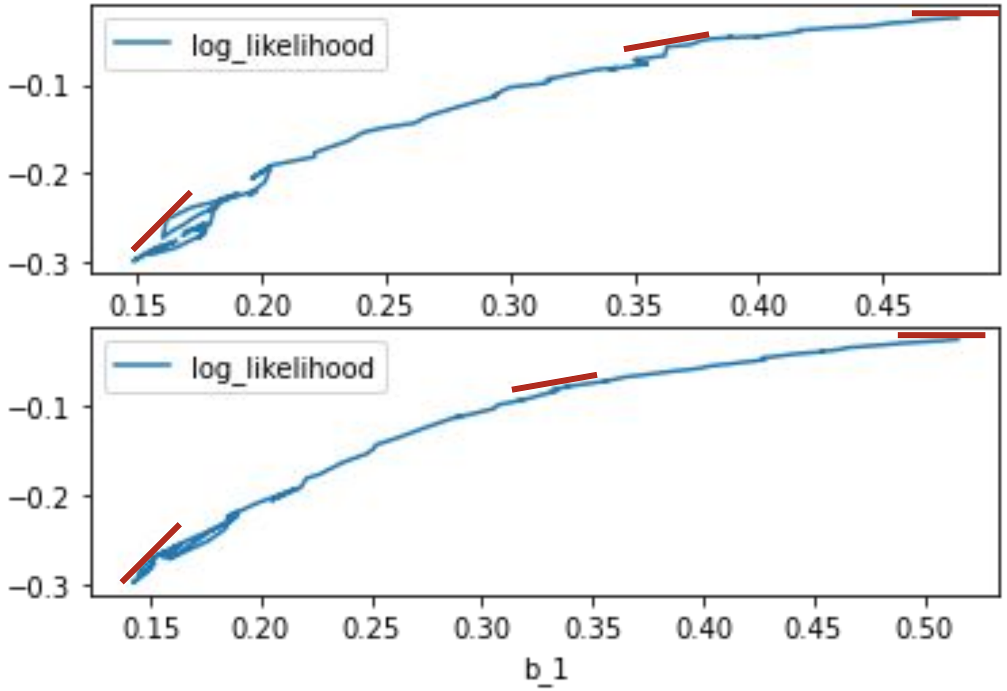
$$= -x_1 - (-x_1)(\frac{1}{1+e^{(b_1x_1+b_2x_2+c)}}) + yx_1$$

$$= -x_1 - (-x_1)(\frac{1}{1+e^{(b_1x_1+b_2x_2+c)}}) + yx_1$$

$$= \frac{-x_1+x_1+-x_1e^{(b_1x_1+b_2x_2+c)}}{1+e^{(b_1x_1+b_2x_2+c)}}) + yx_1$$

$$= x_1(y - \frac{e^{(b_1x_1+b_2x_2+c)}}{1+e^{(b_1x_1+b_2x_2+c)}})$$

$$\frac{dl}{db_1} = x_1(y - \frac{1}{1+e^{-(b_1x_1+b_2x_2+c)}})$$



Iterar sobre los datos para aprender b_0

Algorithm 21 $\textsc{GradientDescent}(\mathcal{F}, K, \eta_1, \ldots)$

$z^{(0)} \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // initialize variable we are optimizing

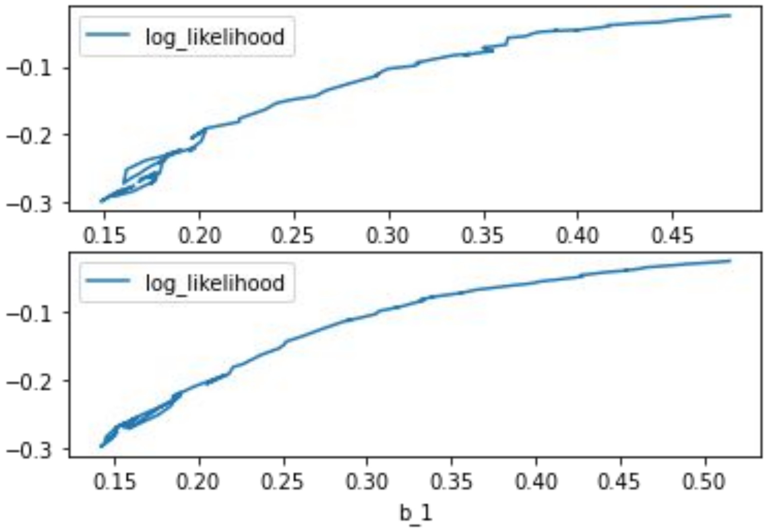
for $k = 1 \ldots K$ **do**

  $g^{(k)} \leftarrow \nabla_z \mathcal{F}|_{z^{(k-1)}}$ // compute gradient at current location

  $z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} g^{(k)}$ // take a step down the gradient

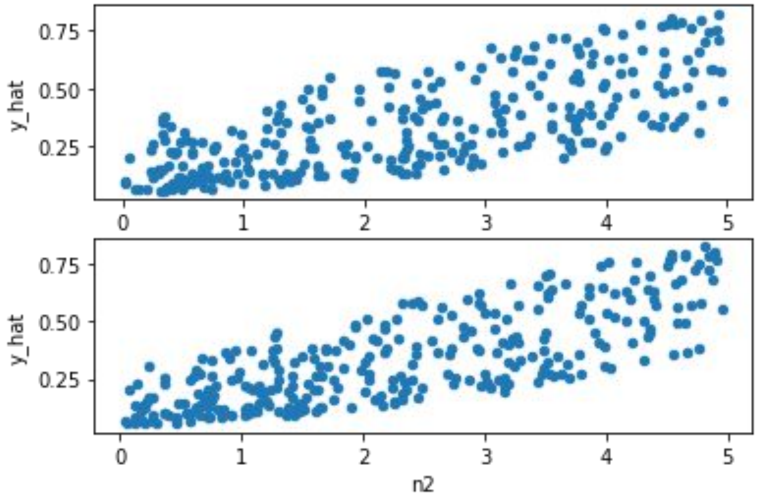
end for

return $z^{(K)}$



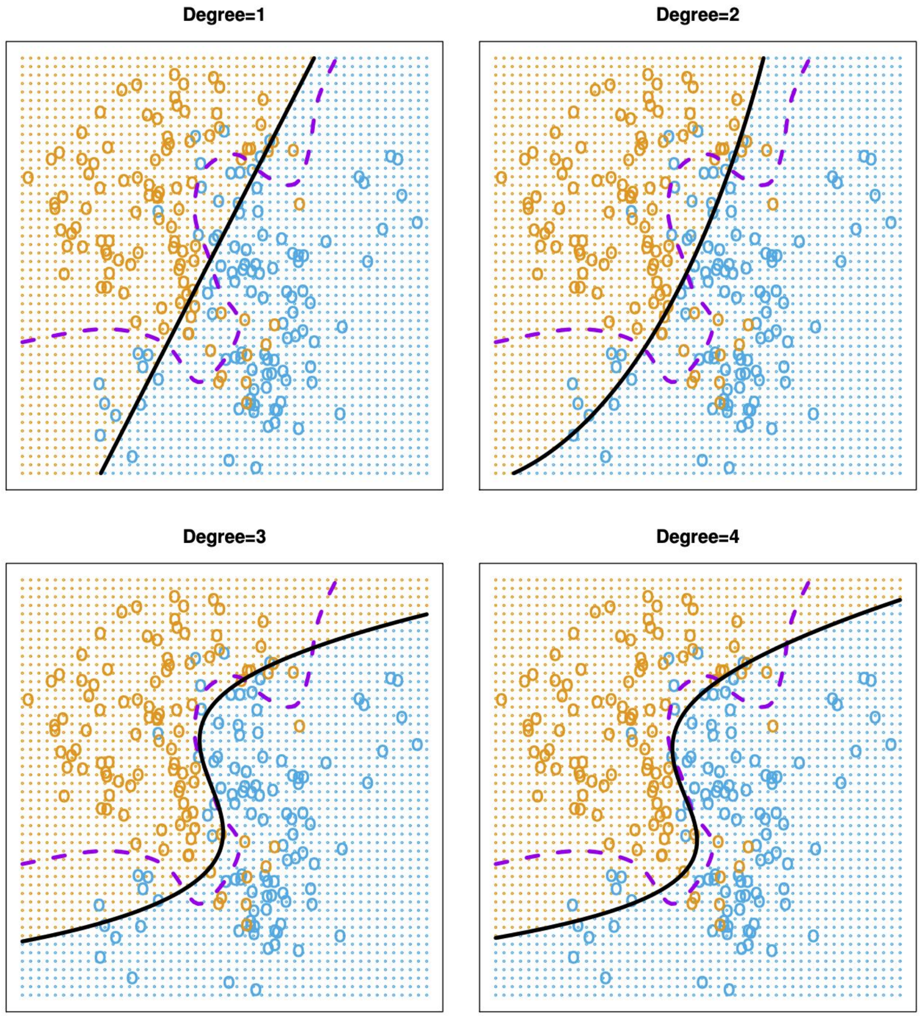
Valores encontrados por el gradiente, en el ejemplo de las notas:
Está muy cerca del valor real

bo_hat: 0.48045085346873073, b_0 = 0.5
b1_hat: 0.5152842933767637, b_1=0.5



* A Course of Machine Learning http://ciml.info/
* Gareth James, An Introduction to Statistical Learning

Recuerde, la regresión logística necesita transformaciones de los datos para mejorar



$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X_2^2.$$

Igual que la regresión lineal, se pueden incluir polinomios para tener más flexibilidad a la hora de dividir los datos.

* A Course of Machine Learning http://ciml.info/
* Gareth James, An Introduction to Statistical Learning

Codigo para hacer la regresión

Similar a la regresión, sólo asumimos las forma logistica

```
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
model = sm.Logit(y_train, X_train).fit(method='bfgs',maxiter=10000)
model.summary()
```