Logistic Regression

like linear regression but used for classification

classification → binary → multi → etc...

① Estimating probability (Sigmoid)

1. it compute weighted sum of features + bias (like linear Regres)

2. then compute logistic of this result ($\hat{p}$)

Note:

$logit(p) = \log\left(\frac{p}{1-p}\right)$

= inverse of logistic

= also called "log-odds

= log ratio positive predicatio prob & neg

$$\hat{p} = h_\theta(x) = \underbrace{\sigma}_{\text{Sigmoid}}(\underbrace{x^T\theta}_{\text{weighted sum + Bias}})$$

- $\sigma(t) = \dfrac{1}{1+e^{-t}}$

- it has range [0, 1]

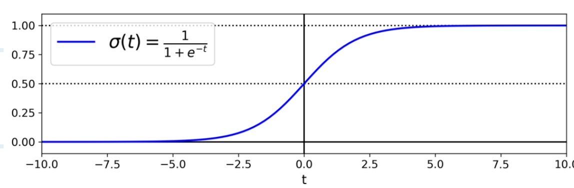


Figure 4-21. Logistic function

- fun fact: called sigmoid as "shape of S"

3. Make predication ($\hat{y}$)

$$\hat{y} = \begin{cases} 1 & \hat{p} \geq 0.5 \rightarrow \text{positive sample} \\ 0 & \hat{p} < 0.5 \rightarrow \text{negative sample} \end{cases}$$

meaning $\hat{y} = \begin{cases} 1 & x^T\theta \geq 0 \\ 0 & x^T\theta < 0 \end{cases}$

4. Calculate Loss Function

$$c(\vec{\theta}) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1-\hat{p}) & \text{if } y=0 \end{cases}$$

to Make sense & @ $y=1 \longrightarrow$ we want $\hat{p}=1 \longrightarrow$ f $-\log(\hat{p})$ $= -\log(1) = 0$

But if it approach zero $\hat{p} \simeq 0$ it go to "∞"

the reverse @ $y=0 \longrightarrow$ we want $\hat{p}=0 \rightarrow -\log(1-\hat{p})$ $= 0$

But if $\hat{p}=1$ Cost $\simeq \infty$

5_ Calculate Cost Function

$$J(\vec{\theta}) = \text{avg of Loss}$$

$$= \frac{-1}{m} \sum_{i=1}^{m} y^{(i)} \log(\hat{p}^{(i)}) + (1-y^{(i)}) \log(1-\hat{p}^{(i)})$$

$$= \frac{-1}{m} \sum_{i=1}^{m} y^{(i)} \log(\underbrace{\hat{p}^{(i)}}_{\sigma(x^T\theta)}) + \bar{y}^{(i)} \log(\bar{\hat{p}}^{(i)})$$

Forward end here

6_ Calculate partial Derivative of Cost

$\because$ $\sigma'(x) = \frac{d}{dx}\sigma(x) = \sigma(x)\,(1-\sigma(x))$

$\because$ $\frac{d}{dx}\log(F(x)) = \frac{F'(x)}{F(x)}$

$\because$ $\frac{d}{dx}F(g(x)) = F'(g(x))\; g'(x) \Longrightarrow$ Chain Rule of derivatives so $\frac{d}{d\theta_i} \log(\sigma(X^T\theta)) = \frac{\sigma'(X^T\theta)}{\sigma(X^T\theta)} = \frac{\sigma(X^T\theta)(1-\sigma(X^T\theta)) \cdot x_i}{\sigma(X^T\theta)}$

$= (1-\hat{p})\, x_i$

$\frac{d}{d\theta_i} \log(1-\sigma(X^T\theta)) = \frac{-(\hat{p})(1-\hat{p})\, x_i}{1-\hat{p}} = -\hat{p}\, x_i$

$\therefore J(\vec{\theta}) = \frac{-1}{m} \sum_{i=1}^{m} y^{(i)} \log(\hat{p}) + (1-y^{(i)}) \log(1-\hat{p})$

so $\frac{d}{d\theta_j} J(\vec{\theta}) = \frac{-1}{m} \sum_{i=1}^{m} y^{(i)} (1 \cancel{-\hat{p}})\, x_j^{(i)} + (1 \cancel{- y^{(i)}}) (-\hat{p}\, x_j^{(i)})$

$= \frac{-1}{m} \sum_{i=1}^{m} (y^{(i)} - \hat{p})\, x_j^{(i)}$

$= \frac{1}{m} \sum_{i=1}^{m} (\sigma(\theta^T x^{(i)}) - y^{(i)})\, x_j^{(i)}$

$\Rightarrow$ it's like linear regression partial derivative

↳ it compute perdication error & multiply it by j^{th} feature

7- Now (Batch, Mini_Batch, Stochastic) Gradient Descent

② Decision Boundry

1-D Visualization

'predict_proba'

unconfident Region

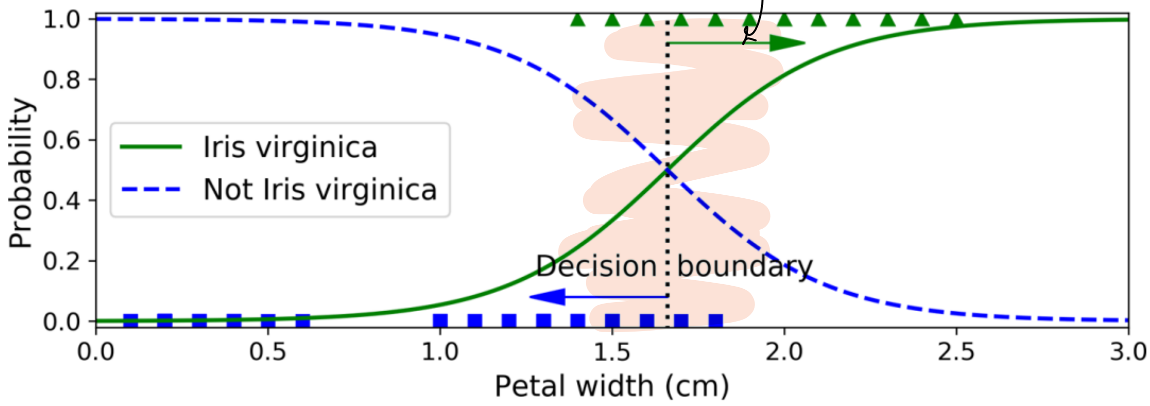
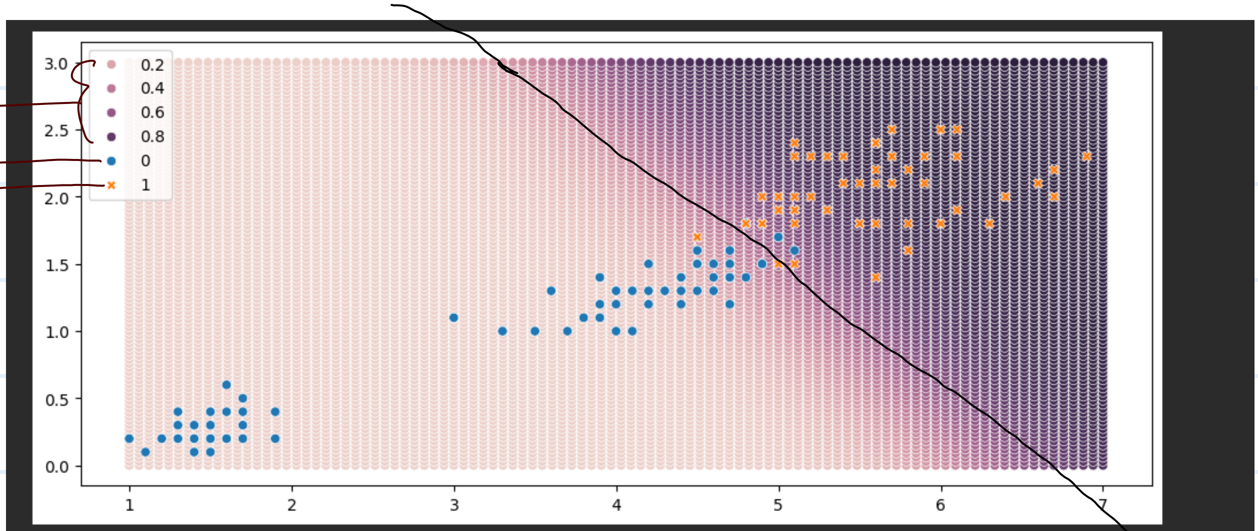


Figure 4-23. Estimated probabilities and decision boundary

2-D Visualization

predict proba

Train data



decision Boundry

Note : Logistic Regression → use "C" which inverse "α" to Control regularization

"implementation" in NoteBook

3) Softmax Regression (multi nomial Regression)

Logistic Regression ⟹ it's used for multi-class / not multi-output

⟹ without train binary Classifier for each class

Steps:

1- Calculate Score function for each class ($s_k(\vec{x})$) for each class K

$$S_k(\vec{x}) = \vec{x}^T \vec{\theta}^{(k)}$$

there's parameters for each class stored as $\Theta = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_k \end{pmatrix}$

2- Estimate probability of each class

2.1- By taking exponential to Score

2.2- then normalize it

$$\hat{P}_k = \sigma(\vec{s(x)})_k = \frac{e^{s_k(\vec{x})}}{\sum_{j=1}^{K} e^{s_j(\vec{x})}}$$

estimated probability to class "k"

K = # Classes

$s(\vec{x})$ = Vector of Scores per each class

3- Predict Class which has ⟹ highest probability

= highest Score

$$\hat{y} = \underset{k}{\text{argmax}}\,(\hat{P}_k) = \underset{k}{\text{argmax}}\,(s_k(\vec{x})) = \underset{k}{\text{argmax}}\,(\theta_k^T \cdot \vec{x})$$

Backward step

4. Cross_Entropy loss

$$J(\Theta) = -\sum_{i=1}^{m} H(p^{(i)}, \hat{p}^{(i)})$$

Where $p^{(i)} = [0\ 0\ 0\ 1\ 0]$ for $y=4$ for 5_classes

$$= \frac{-1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log(\hat{p}_k^{(i)})$$

$$= \frac{-1}{m} \sum_{i=1}^{m} \log(\hat{p}_{(y)}^{(i)})$$

estimated probability of true class

Note : 2.1 Note contain Cross Entropy

5. let's calculate partial derivative

$$\nabla_{\theta^{(k)}} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{p}_k^{(i)} - y_k^{(i)}) \vec{x}^{(i)}$$

6. Update parameters

then use partial derivate to update $\theta_{(k)}$ for each class

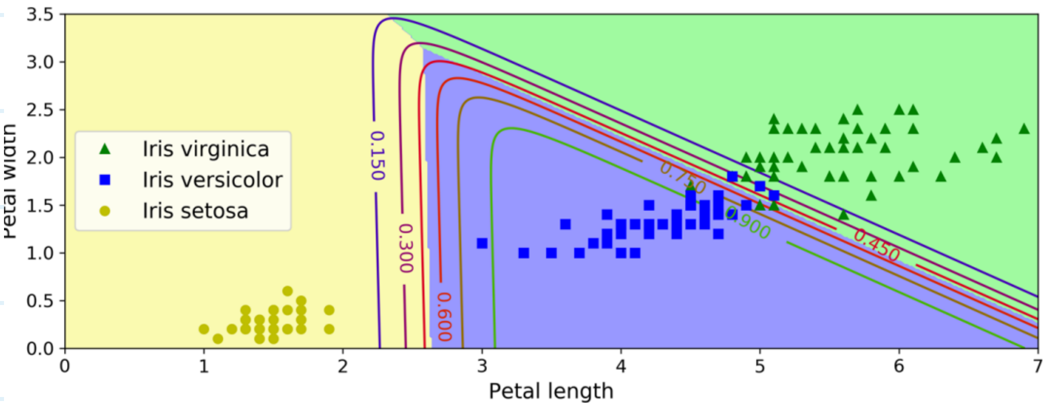


Figure 4-25. Softmax Regression decision boundaries