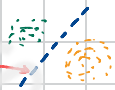
Logistic Regression

Autor: Antonio Castañares

- It's a discriminate classifier (linear classifier), remember classification is a regression problem with discrete labels
- Unlike linear regression, which predicts continuous values, logistic regression predicts probabilities
- The decision boundary is determined by sigmoid function

Types of classifiers:

- Discriminate:
  - Goal: Directly estimate $P(y|x)$
  - What's learned: Decision boundary
  - Examples: Logistic Regression, SVMs



- Generative:
  - Goal: Estimate $P(x|y)$ to then deduce $P(y|x)$
  - What's learned: Probability distribution of data
  - Examples: GDA, Naïve Bayes

- Discriminate: Training data $\xrightarrow[P(h)]{\text{estimate}}$ $P(D|h)$ $\xrightarrow[\text{Bayes)}]{\text{Calculate (using}}$ $P(h|D)$ Naïve Bayes: $Y_{NB} = \underset{h}{\text{argmax}}\,(P(D|h)\cdot P(h))$
- Generative: Training data $\xrightarrow[\text{estimate directly}]{}$ $P(h|D)$ Logistic Regression: $Y_{LR} = \underset{h}{\text{argmax}}\,(P(h|D))$

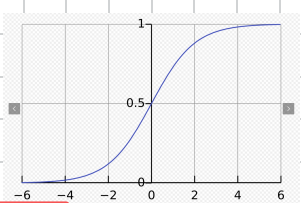
- Idea: Apply linear regression to map predictions to probabilities

Binary classification

Linear regression: $y = wX + b$, predicted values are not guaranteed between 0 and 1

Solution: Apply sigmoid function to linear regression

Sigmoid function => $\sigma(z) = \frac{1}{1 - e^{-z}}$ where $z = wX + b$



Now, the predicted probability of $P(y=1|X)$ is: $P(y=1|X) = \frac{1}{1 + e^{-(wX+b)}}$

- Decision boundary:
  - Class 0: $P(y=1|X) < 0.5$ or $P(y=0|X) \geq 0.5$
  - Class 1: $P(y=1|X) \geq 0.5$ or $P(y=0|X) < 0.5$

- Logistic regression is used for binary classification and multiclass classification
- For binary classification used sigmoid function
- For multiclass classifier used softmax function

Softmax: return a probability distribution over all classes and all output sums to 1

$$\text{Sigmoid} = \frac{1}{1+e^{-z}} \qquad \text{Softmax} = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad \text{where } z = wx+b$$

Example:

SPAM classifier: where $z = 2.5 \rightarrow \frac{1}{1+e^{-2.5}} = 0.92$ (92% the message is spam)

3-class classification: where $z = [2, 1, 0.1]$

$$P(y=1) = \frac{e^2}{e^2+e^1+e^{0.1}} = 0.59$$
$$P(y=2) = \frac{e^1}{e^2+e^1+e^{0.1}} = 0.27$$
$$P(y=3) = \frac{e^{0.1}}{e^2+e^1+e^{0.1}} = 0.14$$

$0.59+0.27+0.14 = 1$ All classes sum to 1

- Two ways to handle bias
  - Explicit Bias term: $p(1|x) = \sigma(wx+b)$ → Parameter that is optimized during training
  - Incorporating bias into the feature vector: Add a constant "1" as a extra feature in x:

$$x = \begin{pmatrix} x \\ 1 \end{pmatrix} \rightarrow P(1|x) = \sigma(wx)$$

(Add a new column: X_bias = np.c_[X, np.ones(m)]; m, n = X.shape)

- Maximum Likelihood Estimation (MLE)
- MLE choose the parameter w that maximizes the likelihood function

MLE assumption:
1. Given independent and identically distributed samples:
$$(x_1, x_2, \dots x_n) \sim P(x_1, x_2, \dots x_n | w)$$
2. Find w_* that makes the observed data most likely
$$w_* = \arg\max_w p(x_1, x_2, \dots x_n | w)$$
3. We can express it as a product:
$$L(w) = p(x_1, x_2, \dots x_n | w) = \prod_{i=1}^{n} P(x_i | w)$$

$w_* = \arg\max_w L(w)$

- However, instead of maximizing a product of probabilities (MLE) we minimize the sum of negative log probabilities (NLL)

Because
- Multiplying many small probabilities leads to numerical underflow
- Optimization is easier when we deals with sums instead of products

Explication: Maximizing the likelihood ($\max_w L(w)$) is the same that maximise the log-likelihood ($\max_w \log(L(w))$) and in mathematical optimization is easier to minimize its negative ($\min_w \log(L(w))$).

Negative log-Likelihood (NLL):

In binary classification: $P(y=1|x,w) = \bar{y} = \sigma(wx) = \frac{1}{1+e^{-wx}}$ ← Probability of $y=1$ given the input x and the model with the parameter w

$P(y=0|x,w) = 1-\bar{y}$

This can be combined in $P(y|x,w) = \bar{y}^{y}(1-\bar{y})^{(1-y)}$

NLL uses the likelihood function: $L(w) = -\log(P(y|x,w))$

$L(w) = -\log(\underbrace{\bar{y}^{y}}_{A}\underbrace{(1-\bar{y})^{(1-y)}}_{B})$

Expanding Using log properties: $\log(AB) = \log A + \log B$, $\log(A^B) = B\log A$

This is the cross-entropy loss function: $\boxed{L(w) = -[y\log(\bar{y}) + (1-y)\log(1-\bar{y})]}$

$L(w) = -y\log(\bar{y}) + (y-1)\log(1-\bar{y})$ $\quad -[1-y] = y-1$

$L(w) = -y\log(\bar{y}) + y\log(1-\bar{y}) - \log(1-\bar{y})$ $\quad -AB+AC = A(C-B)$

$L(w) = y[\log(1-\bar{y}) - \log(\bar{y})] - \log(1-\bar{y})$ $\quad \log A - \log B = \log\frac{A}{B}$

$L(w) = y\log\left(\frac{1-\bar{y}}{\bar{y}}\right) - \log(1-\bar{y})$ $\quad 1-\sigma(z) = \frac{1}{1+e^{z}} = \frac{1-\sigma(z)}{\sigma(z)} = e^{-z}$

$\log e^{-z} = -z$

We want to minimize this function to get the best parameters

$L(w) = -yxw + \log(1+e^{xw})$

↓ we average the loss

$\boxed{L(w) = \frac{1}{N}[-yxw + \log(1+e^{xw})]}$

Finally, we solve it using gradient descent using their gradient

$\nabla L(w) = \nabla\frac{1}{N}[-yxw + \log(1+e^{xw})] = \frac{1}{N}\left[\frac{e^{xw}}{1+e^{xw}}x - yx\right] = \frac{1}{N}(\sigma(xw)-y)x$

Which can be written under the matrix form where $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$

$\boxed{\nabla L(w) = \frac{1}{N}X^T(\sigma(Xw)-y)}$