Logistic Regression

@cambridgespark

CAMBRIDGE SPARK

High level overview

- Introduction to classification
- 10 min Practical ipython notebook - view some data

- Introduction to Logistic regression
- 1.5 hour practical ipython notebook - implement logistic regression

- Introduction to regularisation
- 30 min Practical ipython notebook - regularisation

Classification problems

- We will be working with binary classification
- The output we are fitting is either from class 0 or class 1
- We typically code the response variable y as {0, 1}
- Models will generally output a real value in [0, 1]
- This represents the probability of being in class 1
- A classification can be made (if needed!) by thresholding this value
- A typical threshold to use is 0.5

Hands-on session

classification_data.ipynb
(10 mins)

Logistic Regression

Recall linear regression:

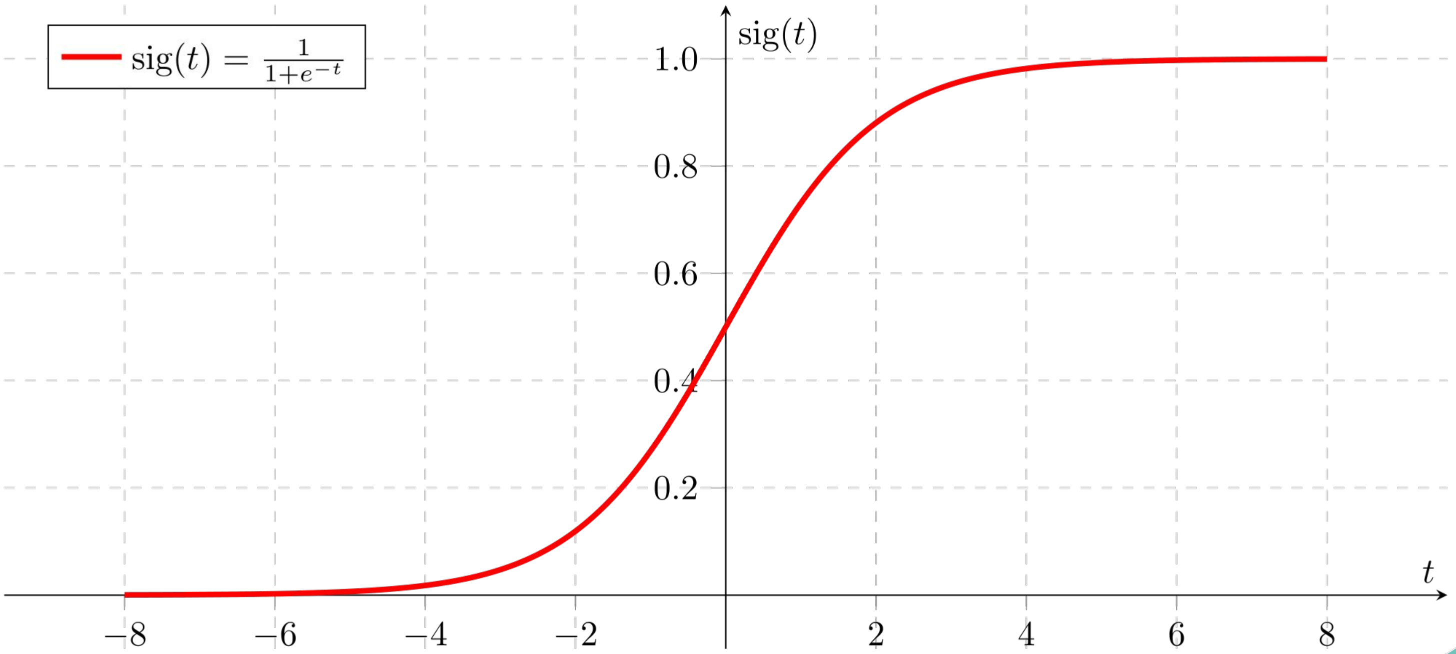
$$f(\boldsymbol{x}) = w_0 x_0 + w_1 x_1 + \ldots + w_D x_D$$
$$= \sum_{i=0}^{D} w_i x_i$$

Logistic regression:

$$f(\boldsymbol{x}) = \sigma(w_0 x_0 + w_1 x_1 + \ldots + w_D x_D)$$
$$= \sigma\left(\sum_{i=0}^{D} w_i x_i\right)$$

The sigmoid function

$$f(t) = \frac{1}{1+e^{-t}}$$



CAMBRIDGE SPARK

Loss function to minimise

if $y =$ true label

i.e. 0 or 1

$p =$ our model output

the predicted probability of class 1

i.e. some real number in [0, 1]

We want something that:

- Returns a large value if p ~ 1 and y = 0, or p ~ 0 and y = 1
  - i.e. the predicted probability is poor
- And returns a small value if prediction is good

CAMBRIDGE SPARK

Cross-entropy

For a single data item:

$$-\log\left(p^y(1-p)^{(1-y)}\right)$$

Summed over the whole data:

$$L = -\sum_{n=1}^{N}\log\left(p_n^{y_n}(1-p_n)^{(1-y_n)}\right)$$

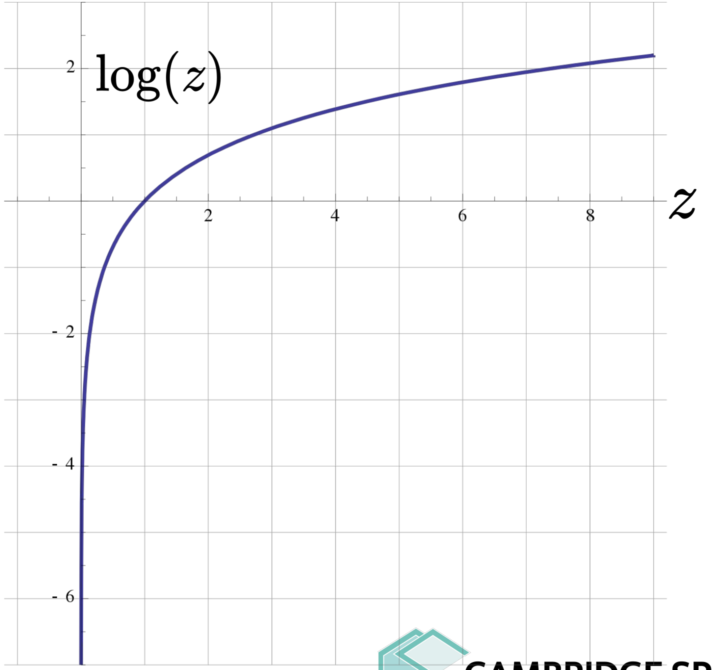
if $y =$ true label

i.e. 0 or 1

$p =$ our model output

the predicted probability of class 1

i.e. some real number in $[0, 1]$



CAMBRIDGE SPARK

Fitting the parameters

for data point $\boldsymbol{x}_n$:

$$p_n = \sigma(w_0 x_{n0} + w_1 x_{n1} + \ldots + w_D x_{nD})$$
$$= \sigma\left(\sum_{i=0}^{D} w_i x_{ni}\right)$$

Our parameters to fit are the w_i

There is no closed form solution

=> gradient descent to minimise loss!

$$L = -\frac{1}{N}\sum_{n=1}^{N} \log\left(p_n^{y_n}\,(1-p_n)^{(1-y_n)}\right)$$

Notice that L is a function of our parameters w_i
(This is because p_n is a function of w)

The 1/N is convention (take average over batch)

CAMBRIDGE SPARK

Gradient of the loss function

Recap from optimisation, gradient descent:

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \gamma \nabla L(\boldsymbol{w}^{(t)})$$

$$\nabla L(\boldsymbol{w}) = [\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial x_D}]$$

$$\gamma = \text{"learning rate" or "step size"}$$

Gradient of the cross-entropy loss is (for one data point):

$$\nabla L(\boldsymbol{w}) = \boldsymbol{x}_n (p_n - y_n)$$

And averaging over all data points:

$$\nabla L(\boldsymbol{w}) = \frac{1}{N} \boldsymbol{X}^T (\boldsymbol{p} - \boldsymbol{y})$$

CAMBRIDGE SPARK

Hands-on session

logistic_regression.ipynb
(1.5 hours)

Problems with logistic regression

- Overconfidence:
  - If your classes are separable, parameters get larger and larger
  - Recall from the notebook: large parameters -> sharp decision boundary
- Outliers
  - An incorrectly labeled datapoint can have a disproportionately large effect on the loss
- Multicolinearity:
  - Variables that are nearly the same everywhere can cause numerical instability
  - E.g. small input changes can cause large output change
  - If goal is to interpret model parameters after fit - this becomes invalid!

Regularisation

$$L(w) = -\frac{1}{N}\sum_{n=1}^{N}\log\Big(\sigma(wx_n)^{y_n}(1-\sigma(wx_n))^{(1-y_n)}\Big)$$

$$w^* = \arg\min_{w} \quad \{L(w) + \lambda||w||_2\}$$

- A simple way to stabilise the model fit is to regularise the parameters
- Typically we simply add a term to the loss function
- For example, L1 simply adds $||w||_1$ to the loss
  - The parameters w are encouraged to have a small L1 distance from the origin
  - Results in pushing as many $w_i = 0$ as possible
  - Possible to have some large w_i so long as many = 0
- L2 add $||w||_2$ to the loss
  - The parameters w are encouraged to have a small L2 distance from the origin
  - Results in making all w_i as close to zero as possible
  - Very hard to have any large w_i , most go to a small nonzero value
- Control how much regularisation with a hyperparameter: λ

CAMBRIDGE SPARK

Hands-on session

Regularisation.ipynb
(30 mins)

CAMBRIDGE SPARK

Further Reading

- http://neuralnetworksanddeeplearning.com/chap3.html
  - Explains derivative of the gradient
  - Extends to multiclass
- Quick reference: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

CAMBRIDGE SPARK