Machine Learning 1.07: Logistic Regression & Polling

Tom S. F. Haines
T.S.F.Haines@bath.ac.uk

UNIVERSITY OF BATH

Logistic Regression

(also called logit regression)

- Does binary classification (non-binary variants).
- But so does a random forest.
- So which is better?

Logistic Regression

(also called logit regression)

- Does binary classification (non-binary variants).
- But so does a random forest.
- So which is better?

- A random forest (mostly).

Logistic Regression

(also called logit regression)

- Does binary classification (non-binary variants).
- But so does a random forest.
- So which is better?

- A random forest (mostly).

- So why should you care?
  - Faster, and can be trained incrementally/online
  - Really popular everywhere but computer science!
  - Simple (Invented in 1958!)

Logistic Regression

(also called logit regression)

- Does binary classification (non-binary variants).
- But so does a random forest.
- So which is better?

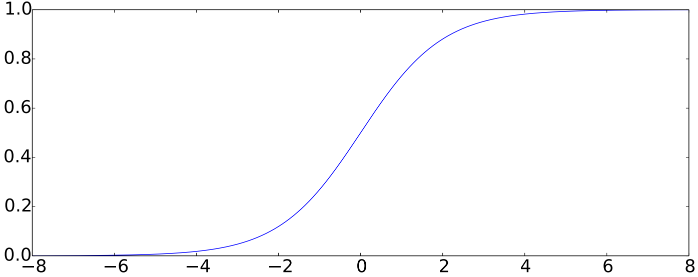
- A random forest (mostly).

- So why should you care?
  - Faster, and can be trained incrementally/online
  - Really popular everywhere but computer science!
  - Simple (Invented in 1958!)

  - Its good for understanding all the subtle details...

The Logistic Function

(also called the sigmoid function)



$$y = \frac{1}{1+e^{-x}} \quad \text{or} \quad y = \frac{e^x}{1+e^x} \quad \text{or} \quad y = \frac{1}{2}\left(1+\tanh(x/2)\right)$$

Maps $-\infty$ – $+\infty$ to $0 - 1$

Fitting To Data I

- Binary classification: Output, y, is 0 or 1.
- For now assume single input feature, x.
- Probabilities go from 0 to 1, so lets assume

$$P(y=1|x) = \frac{1}{1+e^{-z}}, \qquad P(y=0|x) = 1 - \frac{1}{1+e^{-z}} = \frac{e^{-z}}{1+e^{-z}}, \qquad z = \beta_0 + \beta_1 x$$

Fitting To Data I

- Binary classification: Output, y, is 0 or 1.
- For now assume single input feature, x.
- Probabilities go from 0 to 1, so lets assume

$$P(y=1|x) = \frac{1}{1+e^{-z}}, \qquad P(y=0|x) = 1 - \frac{1}{1+e^{-z}} = \frac{e^{-z}}{1+e^{-z}}, \qquad z = \beta_0 + \beta_1 x$$

- This is a **model**.
- β_0 and β_1 are parameters which we fit to data.
- Note the assumptions!

Fitting To Data II

- We have a data set: y_i and x_i for $i \in 1..N$
- We want to maximise the probability of the data set:

$$\underset{\beta_0,\beta_1}{\operatorname{argmax}} \left\{ \prod_{i=1}^{N} P(y_i | x_i, \beta_0, \beta_1) \right\}$$

- No analytic solution.
- Gradient descent!

Fitting To Data II

- We have a data set: y_i and x_i for $i \in 1..N$
- We want to maximise the probability of the data set:

$$\underset{\beta_0,\beta_1}{\operatorname{argmax}} \left\{ \prod_{i=1}^{N} P(y_i | x_i, \beta_0, \beta_1) \right\}$$

- No analytic solution.
- Gradient descent!

- Note:
  - Historically, Logistic regression was optimised by minimising distance.
  - This has a closed form expression, but is idiotic: Output not probabilities.
  - Many implementations still use this approach however. Check!

Simplifying I

- Swap $P()$ for logistic equation:

$$\underset{\beta_0,\beta_1}{\operatorname{argmax}} \left\{ \prod_{i=1}^{N} p_i^{y_i} (1 - p_i)^{1-y_i} \right\}, \qquad p_i = \frac{1}{1 + e^{-z_i}}, \quad z_i = \beta_0 + \beta_1 x_i$$

Simplifying I

- Swap $P()$ for logistic equation:

$$\underset{\beta_0,\beta_1}{\operatorname{argmax}} \left\{ \prod_{i=1}^{N} p_i^{y_i} (1-p_i)^{1-y_i} \right\}, \qquad p_i = \frac{1}{1+e^{-z_i}}, \quad z_i = \beta_0 + \beta_1 x_i$$

- Take the log – argmax doesn't care:

$$\underset{\beta_0,\beta_1}{\operatorname{argmax}} \left\{ \sum_{i=1}^{N} y_i \log(p_i) + (1-y_i) \log(1-p_i) \right\}$$

UNIVERSITY OF BATH

Simplifying I

- Swap $P()$ for logistic equation:

$$\underset{\beta_0,\beta_1}{\operatorname{argmax}}\left\{\prod_{i=1}^{N} p_i^{y_i}(1-p_i)^{1-y_i}\right\}, \qquad p_i = \frac{1}{1+e^{-z_i}}, \quad z_i = \beta_0 + \beta_1 x_i$$

- Take the log – argmax doesn't care:

$$\underset{\beta_0,\beta_1}{\operatorname{argmax}}\left\{\sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log(1-p_i)\right\}$$

- Rearrange:

$$\underset{\beta_0,\beta_1}{\operatorname{argmax}}\left\{\sum_{i=1}^{N} \log(1-p_i) + \sum_{i=1}^{N} y_i \log\left(\frac{p_i}{1-p_i}\right)\right\}$$

UNIVERSITY OF BATH

Simplifying II

- What is $\log(1 - p_i)$?

$$\log\left(1 - \frac{1}{1 + e^{-z_i}}\right) = \log\left(\frac{1}{1 + e^{z_i}}\right) = -\log(1 + e^{z_i})$$

Simplifying II

- What is $\log(1-p_i)$?

$$\log\left(1-\frac{1}{1+e^{-z_i}}\right)=\log\left(\frac{1}{1+e^{z_i}}\right)=-\log(1+e^{z_i})$$

- What is $\log\left(\frac{p_i}{1-p_i}\right)$?

$$\log\left(\frac{1}{1+e^{-z_i}}\frac{1+e^{-z_i}}{e^{-z_i}}\right)=z_i$$

UNIVERSITY OF BATH

Simplifying II

- What is $\log(1 - p_i)$?

$$\log\left(1 - \frac{1}{1 + e^{-z_i}}\right) = \log\left(\frac{1}{1 + e^{z_i}}\right) = -\log(1 + e^{z_i})$$

- What is $\log\left(\frac{p_i}{1-p_i}\right)$?

$$\log\left(\frac{1}{1 + e^{-z_i}} \frac{1 + e^{-z_i}}{e^{-z_i}}\right) = z_i$$

- So...

$$\underset{\beta_0, \beta_1}{\operatorname{argmax}} \left\{ \sum_{i=1}^{N} y_i z_i - \sum_{i=1}^{N} \log(1 + e^{z_i}) \right\}$$

Gradient

- For β_j ($L =$ likelihood, i.e. term in argmax):

$$\frac{\delta L}{\beta_j} = \frac{\delta L}{\delta z}\frac{\delta z}{\beta_j} = \sum_{i=1}^{N} \left\{ \left(y_i - \frac{1}{1 + e^{-z_i}} \right) \frac{\delta z_i}{\delta \beta_j} \right\}$$

Gradient

- For β_j ($L =$ likelihood, i.e. term in argmax):

$$\frac{\delta L}{\beta_j} = \frac{\delta L}{\delta z}\frac{\delta z}{\beta_j} = \sum_{i=1}^{N}\left\{\left(y_i - \frac{1}{1+e^{-z_i}}\right)\frac{\delta z_i}{\delta \beta_j}\right\}$$

- β_0:

$$\frac{\delta L}{\beta_1} = \sum_{i=1}^{N}\left\{\left(y_i - \frac{1}{1+e^{-z_i}}\right)\right\}$$

- β_1:

$$\frac{\delta L}{\beta_1} = \sum_{i=1}^{N}\left\{\left(y_i - \frac{1}{1+e^{-z_i}}\right)x_i\right\}$$

Model Fitting – Gradient Ascent

- Initialise all β to 0; $\beta_j^{(0)} = 0$.
- Iterate updating each β_j with

$$\beta_j^{(t+1)} = \beta_j^{(t+1)} + \lambda \frac{\delta L}{\beta_j}$$

and λ set to a suitably small step size.

- Stop when the β values stop changing / L stops increasing.

Multiple Features

- Replace z_i with:

$$z_i = \vec{\beta} \cdot \vec{x_i}$$

  where

$$\vec{\beta} = [\beta_0, \beta_1, \beta_2, \ldots]^T \qquad \vec{x_i} = [1, x_{i,1}, x_{i,2}, \ldots]^T$$

- All of the above just works, exactly as you would expect!

Probability

- Why do we care about getting probability?

Probability

- Why do we care about getting probability?
  - Communicating uncertainty to users.
  - Communicating uncertainty to the “next step”.

Probability

- Why do we care about getting probability?
    - Communicating uncertainty to users.
    - Communicating uncertainty to the "next step".

- Are we getting probability?

UNIVERSITY OF BATH

Probability

- Why do we care about getting probability?
  - Communicating uncertainty to users.
  - Communicating uncertainty to the “next step”.

- Are we getting probability?
  - Maybe. . .
  - Optimising a reasonable metric (maximum likelihood).

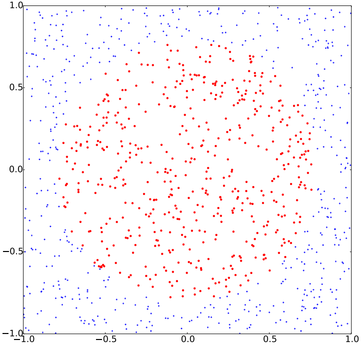
  - But assuming a very specific distribution.
  - Few data sets follow this.

Probability

- Why do we care about getting probability?
  - Communicating uncertainty to users.
  - Communicating uncertainty to the "next step".
- Are we getting probability?
  - Maybe. . .
  - Optimising a reasonable metric (maximum likelihood).
  - But assuming a very specific distribution.
  - Few data sets follow this.
- Are random forests better?

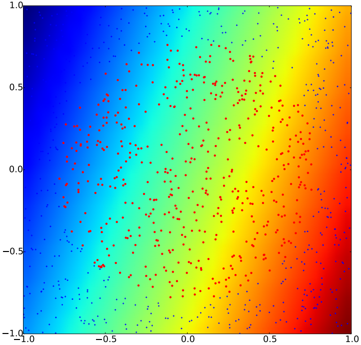
Straight in Feature Space

Can't separate with a straight line:



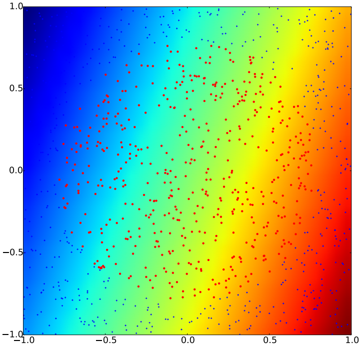
Straight in Feature Space

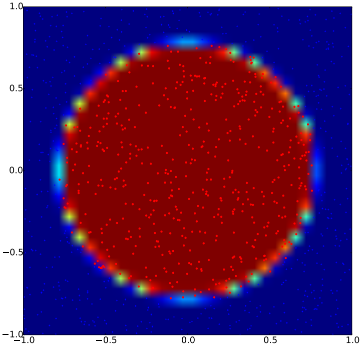
Can't separate with a straight line:



Straight in Feature Space

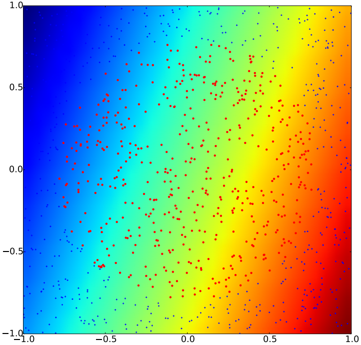
Can't separate with a straight line:

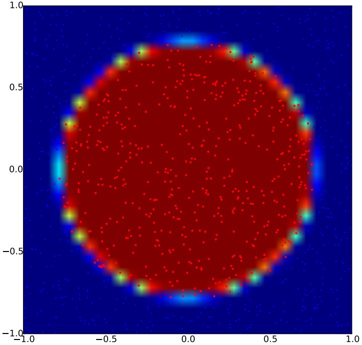




Straight in Feature Space

Can't separate with a straight line:





- Added a third feature - distance from centre of circle.
- A straight line in **feature space** can now separate the data.

Feature Engineering I

- Advanced machine learning: Designs features for you.
- Simple machine learning: You have to design them yourself.

Feature Engineering I

- Advanced machine learning: Designs features for you.
- Simple machine learning: You have to design them yourself.

- In some cases you can't beat the fully automatic approach.
- But in many a simple algorithm with the right features will win!

Feature Engineering I

- Advanced machine learning: Designs features for you.
- Simple machine learning: You have to design them yourself.

- In some cases you can't beat the fully automatic approach.
- But in many a simple algorithm with the right features will win!

- Compute time is cheaper than your time.
- Automatic feature design is taking over.

Feature Engineering II

Examples:

- Convert a continuous variable into a set of discrete variables, marking which bin of a histogram that continuous variable has landed in.

- As above, but with triangular kernels – this is equivalent to letting the algorithm choose an arbitrary piecewise linear function of a feature.

- Calculate random non-linear functions of the input features (Reservoir computing).

Use case: Polling I

Goal: Guess which way an election is going to go.

Use case: Polling I

Goal: Guess which way an election is going to go.

- Sample population (telephone, internet), ask questions:
  - Voting intention
  - Demographics (age, gender etc.)
  - Other useful stuff

  Problem: Can't sample more than a fraction of a percent of population.

Use case: Polling I

Goal: Guess which way an election is going to go.

- Sample population (telephone, internet), ask questions:
    - Voting intention
    - Demographics (age, gender etc.)
    - Other useful stuff

  Problem: Can't sample more than a fraction of a percent of population.

- Idiots approach – simulate election with random sample.
- Will not work as sample too small.

Use case: Polling I

Goal: Guess which way an election is going to go.

- Sample population (telephone, internet), ask questions:
  - Voting intention
  - Demographics (age, gender etc.)
  - Other useful stuff

  Problem: Can't sample more than a fraction of a percent of population.

- Idiots approach – simulate election with random sample.
- Will not work as sample too small.

- Normal approach (most polling companies):
  - Re-weight sample members, e.g. if census says 10% of the country are managers, but it's only 5% in sample double the value of their vote (called raking).
  - Simulate election with this correction.
  - Add fudge factors to ensure you would have got it right last time!

Use case: Polling II

- Smart approach (used by YouGov and FiveThirtyEight):
  - Data set 1: Polling data.
  - Data set 2: Entire country: Last census plus anything else available!
  - Learn function using polling data:
    - Input: Questions shared between between two data sets.
    - Output: How the person is going to vote.
  - Simulate election for **entire country**!

UNIVERSITY OF BATH

Use case: Polling II

- Smart approach (used by YouGov and FiveThirtyEight):
  - Data set 1: Polling data.
  - Data set 2: Entire country: Last census plus anything else available!
  - Learn function using polling data:
    - Input: Questions shared between between two data sets.
    - Output: How the person is going to vote.
  - Simulate election for **entire country**!

- This is the **multilevel regression and poststratification** algorithm.
- Typically called "Mr P".

Use case: Polling II

- Smart approach (used by YouGov and FiveThirtyEight):
  - Data set 1: Polling data.
  - Data set 2: Entire country: Last census plus anything else available!
  - Learn function using polling data:
    - Input: Questions shared between between two data sets.
    - Output: How the person is going to vote.
  - Simulate election for **entire country**!

- This is the **multilevel regression and poststratification** algorithm.
- Typically called “Mr P”.

- “multilevel”: Include terms to account for group biases (in this context).
- “poststratification” means running per-voting region with region demographics.
- Instead of fudge factors ask how people voted last time!

Related models

- Probit regression: Replace the logistic distribution with the Gaussian distribution.
- Whole family of these: Generalised Linear Model.

- Hidden and output nodes in a neural network may each be logistic regression.
- Used to be the dominant choice, these days linear rectified units are preferred to sigmoid functions.

End.

- Logistic regression
- Probability - not simple!
- Basic feature engineering

Further Reading

- Guide to Mr P:
  http://www.princeton.edu/~jkastell/MRP_primer/mrp_primer.pdf
- Andrew Gelman, inventor of Mr P, has an excellent blog: http://andrewgelman.com/
  (his take downs of bad research are particularly educational)