COMP30027 Machine Learning
Logistic Regression

Semester 1, 2019

Afshin Rahimi & Jeremy Nicholson & Tim Baldwin
& Karin Verspoor

THE UNIVERSITY OF
MELBOURNE

Back to classification

- In linear regression, we have continuous features and a continuous output variable.
- In a classification setting, we don't have a continuous output variable, we have discrete classes.

Regression $\leftrightarrow$ Classification I

We have a continuous class, but a discrete learner ("classifier"):

- Map continuous values onto discrete labels: **discretisation**
  - set range of continuous variable that corresponds to each discrete class
- (The data for Project 2 is set up a little like this.)

Regression $\leftrightarrow$ Classification II

We have a discrete class, but a continuous learner ("regressor"):

- Build a suite of regression tasks via **multi-response linear regression**:
  - perform one regression per discrete class, with all instances of that class set to 1, and other instances set to 0
  - classify a given test instance by regressing its value relative to each class, and selecting the class with the highest value
- Somewhat similar to "one-vs-rest" multi-class SVM
- Clumsy, but suggestive of a better solution...

Probabilistic classification I

Remember Naive Bayes?

$$\hat{c} = \arg\max P(c|T)$$

We said that we couldn't solve this directly...
But lots of different values can be regressed!

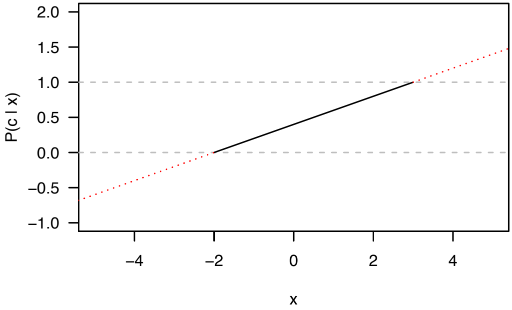
Probabilistic classification II

- Probabilistic classification: attempt to model $P(c|T)$ directly
- Assuming a 2-class (Y/N) problem:
  - Every training instance I with class = Y: $P(c = Y|I) = 1$
  - Every training instance J with class = N: $P(c = Y|J) = 0$
- (Numerical) attributes in the training instances are now predictors; the (numerical) probability is the target quantity
- Assuming linear regression:

$$\begin{aligned} \hat{P}(c = Y|\boldsymbol{x}) &= \beta \cdot x \\ &= \beta_0 + \beta_1 x_1 + \ldots \beta_D x_D \end{aligned}$$

Probabilistic classification III

But no guarantee that $\hat{P} \in [0, 1]$ — is this a problem?



Log-linear models I

Instead of linear regression, consider the following (multiplicative) formulation:

$$
\begin{aligned}
P(c|\boldsymbol{x}) &= \gamma_0 \cdot \gamma_1^{x_1} \cdot \ldots \cdot \gamma_D^{x_D} \\
\log P(c|\boldsymbol{x}) &= \log \gamma_0 + x_1 \log \gamma_1 + \ldots + x_D \log \gamma_D
\end{aligned}
$$

- This looks a little more like NB (summing log probabilities)
- Exponentiation is particularly suited to $\boldsymbol{x}$ representing event frequencies
  - $\leftarrow$ follows from multinomial distribution [probability mass function: $\frac{n!}{x_1! \ldots x_D!} p_1^{x_1} \ldots p_D^{x_D}$]

Log-linear models II

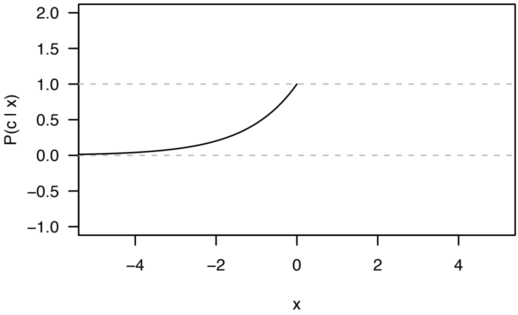
Instead of linear regression, consider the following (multiplicative) formulation:

$$\begin{aligned}
P(c|\boldsymbol{x}) &= \gamma_0 \cdot \gamma_1^{x_1} \cdot \ldots \cdot \gamma_D^{x_D} \\
\log P(c|\boldsymbol{x}) &= \log \gamma_0 + x_1 \log \gamma_1 + \ldots + x_D \log \gamma_D \\
\log P(c|\boldsymbol{x}) &= \beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D
\end{aligned}$$

- Directly fit $\beta_i \equiv \log \gamma_i$
- ("lots of different values can be regressed!")

$\log(P) = \beta x$

$$
\begin{aligned}
\log(P) &= \beta x \\
P &= e^{\beta x}
\end{aligned}
$$



- Curve has unbalanced shape:
  - Fine granularity of response as $P \rightarrow 0$
  - Coarse response as $P \rightarrow 1$
  - $\beta \cdot \boldsymbol{x} > 0 \rightarrow P > 1$

Balanced in P

- Want behaviour that is same for high P and low P
- This is provided by *log odds* or *logit*:

$$\begin{aligned}\mathrm{logit}(P) &= \log \frac{P}{1-P} \\ \mathrm{logit}(1-P) &= -\mathrm{logit}(P)\end{aligned}$$

Logistic regression

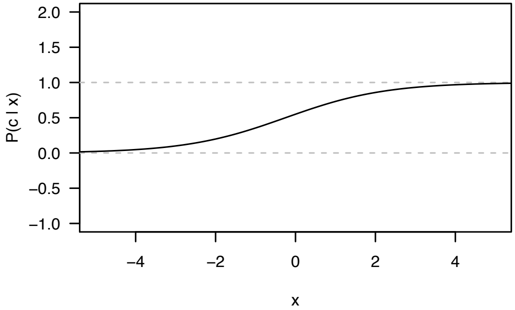
Putting this together, we get:

$$\begin{aligned}
\operatorname{logit} P(c|\boldsymbol{x}) = \log \frac{P(c|\boldsymbol{x})}{1 - P(c|\boldsymbol{x})} &= \beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D \\
P(c|\boldsymbol{x}) &= \frac{1}{1 + \mathrm{e}^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D)}}
\end{aligned}$$

- Expression on rhs of this equation known as logistic function
- So this is called logistic regression

Logistic function I

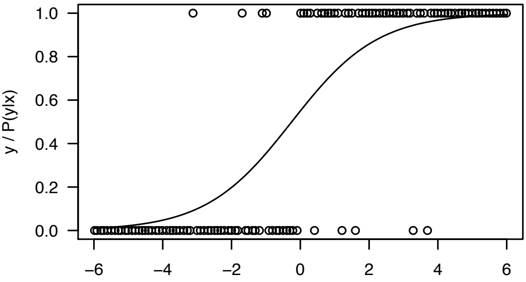
$$P(c|\boldsymbol{x}) = \frac{1}{1 + e^{-(\beta x)}}$$



- $P \in [0, 1]$ for all $\beta \cdot \boldsymbol{x}$

Logistic function II

- Most values for $\beta \cdot \boldsymbol{x}$ lead to $P \approx 1$ or $P \approx 0$
  - This is intended behaviour, because all instances are labelled with 1 or 0
- When predicting:
  - Positive $\beta \cdot \boldsymbol{x}$ means class is Y
  - Negative $\beta \cdot \boldsymbol{x}$ means class is N
  - $\beta \cdot \boldsymbol{x} \approx 0$ means most uncertainty



Logistic Regression classification

- Recall Naive Bayes classification where we model $P(c_j|\boldsymbol{x})$:

$$c = \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i|c_j)$$

- In Logistic Regression, we model $P(c_j|x_1, x_2, ..., x_D)$ directly (subject to parameter β); no need to estimate $P(\boldsymbol{x}|c)$

$$\begin{aligned} & P(c_j|x_1, x_2, ..., x_D; \beta) \\ = & \ \mathrm{logistic}(\beta \cdot \boldsymbol{x}) = \frac{1}{1 + \mathrm{e}^{-(\beta \cdot \boldsymbol{x})}} \\ = & \ h_\beta(\boldsymbol{x}) \end{aligned}$$

Training a Logistic Regression model I

How do we determine β?

- Gradient Descent!
- ... But we need an error function.

Training a Logistic Regression model II

- Our goal is to choose β, so that:
  - the probability $P(y = 1|\boldsymbol{x}) = h_\beta(\boldsymbol{x})$ is close to 1, when x belongs to the Y class
  - the probability $h_\beta(\boldsymbol{x})$ is close to 0, when x belongs to the N class

$$P(y = 1|\boldsymbol{x}; \beta) = h_\beta(\boldsymbol{x})$$
$$P(y = 0|\boldsymbol{x}; \beta) = 1 - h_\beta(\boldsymbol{x})$$
$$\rightarrow \quad P(y|\boldsymbol{x}; \beta) = (h_\beta(\boldsymbol{x}))^y * (1 - h_\beta(\boldsymbol{x}))^{1-y}$$

Training a Logistic Regression model III

- But, we have many (independent) training instances (N)!
- We want to choose β to maximise the **likelihood** of observing our training instances

$$
\begin{aligned}
P(Y|\boldsymbol{X};\beta) &= \prod_{i=1}^{N} P(y_i|\boldsymbol{x_i};\beta) \\
&= \prod_{i=1}^{N} (h_\beta(\boldsymbol{x_i}))^{y_i} * (1 - h_\beta(\boldsymbol{x_i}))^{1-y_i}
\end{aligned}
$$

Training a Logistic Regression model IV

- Or, for simplicity, maximise the **log-likelihood**:

$$\log P(Y|\boldsymbol{X};\beta) \;=\; \sum_{i=1}^{N} y_i \log h_\beta(\boldsymbol{x_i}) + (1-y_i)\log(1-h_\beta(\boldsymbol{x_i}))$$

- This ... looks ugly, but actually has nice derivatives (wrt β_k)
- Consequently, Gradient **Ascent** is the preferred strategy for choosing β

Multi-class classification I

So far, we have only considered 2-class problems:

- Multi-class logistic regression:
  - Take one class ("pivot")
  - For every other class ("Y"), build regression model compared to pivot class ("N")
  - End up with $(|C| - 1)$ different logistic regression models
  - Predict according to the one that has the highest score
- Choice of pivot is hopefully irrelevant (assumes comparison with irrelevant "pivot" class doesn't affect preference between most-likely and second-most-likely classes)

Multi-class classification II

- We need to ensure that the probability distribution for each instance sums to 1:

$$P(y = j|x; \beta) = \frac{\exp(\beta_j \cdot x)}{1 + \sum_{k=1}^{|C|-1} \exp(\beta_k \cdot x)}$$

- Probability of the "pivot" class has 1 in the numerator
- (Log-linear probabilities now need to subtract an unfortunate normalising term for **softmax**)

$$Z_i = \sum_{k=1}^{|C|} e^{\beta_k \cdot \boldsymbol{x_i}}$$

- (Constant with respect to y_i, but not with respect to β_k, which means that it's messy to estimate...)

Logistic Regression Pros/Cons

- Pros:
  - Vast improvement on Naive Bayes
  - Particularly suited to frequency-based features (so, popular in NLP)
- Cons:
  - Slow to train
  - Some feature scaling issues
  - Often needs a lot of data to work well
  - Regularisation a nuisance, but important since overfitting can be a big problem

Summary

- How is **logistic regression** related to regression?
- What is $h_\beta(\boldsymbol{x})$, and:
    - how is it related to probability?
    - how is it used in **gradient ascent**?
- How can logistic regression be (somewhat painfully) extended to multi-class classification?