Introduction to Machine Learning (by Implementation)

Lecture 4: Logistic Regression

Ian J. Watson

University of Seoul

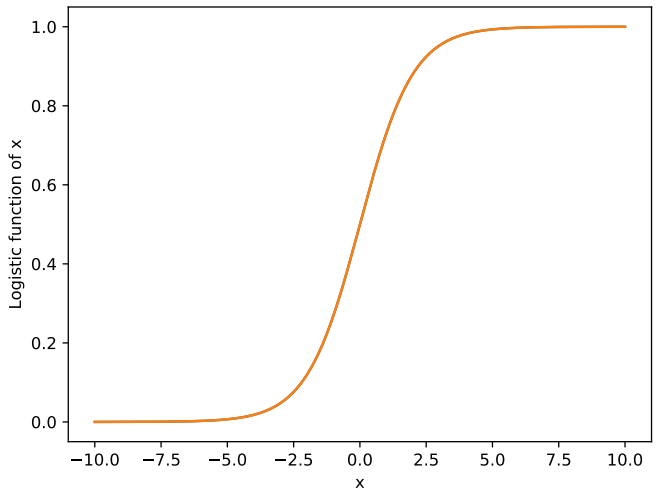
University of Seoul Graduate Course 2019

KRF KOREA RESEARCH FELLOWSHIP
해외우수신진연구자 유치사업

Classification

- We've looked at regression, finding
- Today, we'll look at classification
  - In classification we have some data which could belong to one of several classes
  - We have some well-known data and we want to train a *classifier* which will tell us what class some new, unknown data comes from
  - E.g. classify images of pets into dogs and cats
  - Classify energy depositions in a calo into photons and electrons
  - Based on several indicators (age, height, weight, etc.) say whether someone will get diabetes
- We will learn a basic technique called *logistic regression*
  - Logistic "regression" is classification in disguise

Logistic Function

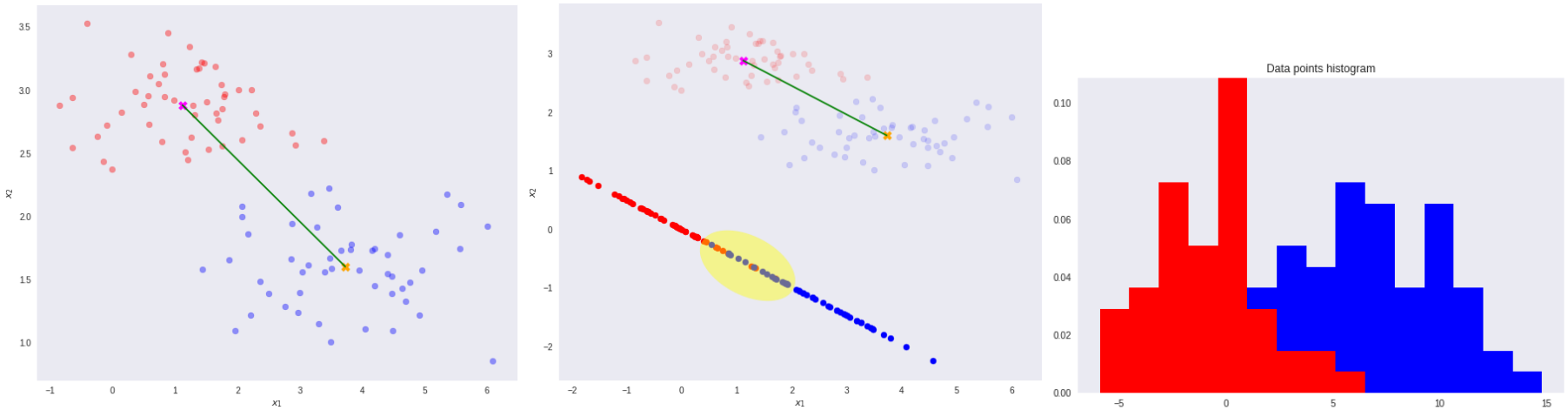


- The logistic function is defined as $f(x) = \frac{1}{1+e^{-x}}$
  - Looks like a classic "turn-on" curve
- "Logistic regression" fits this function from several variables
- Concentrate on the case of two classes (cat/dog or electron/photon), and ask what we want from a classifier output
  - We need to distinguish between the two classes using the output:
  - If the value is 0, it represents the classifier identifying one class (cat)
  - If its near 1, the classifier is identifies the other class (dog)
  - Thus, we need to transform the input variables into 1D, then pass through the logistic function

Logistic regression

- Setup: we have data from two different classes, which can be described by the same independent variables, and we want to distinguish them based on those independent variables
- We want to build a function such that data from one class goes close to 1, from the other close to 0
- We will build a linear function of the variables, then pass it through the logistic function, and try to minimise the distance of data from 0 (for one class), or 1 (for the other)
- $y_i = f(\vec{\beta} \cdot \vec{x_i}) + \epsilon_i$, $y_i = 0$ if x_i from class 0, 1 if x_i from class 1
  - $\vec{\beta} \cdot \vec{x_i} = \beta_0 + \beta_1 x_1 + \ldots \beta_k x_k$ [use functions from last week!]
  - $f(x) = \frac{1}{1+e^{-x}}$ the logistic function
- We will use SGD to find the values of β which minimizes the log-likelihood

Illustration: 1D Projection



- $\vec{\beta} \cdot \vec{x}$ is a projection of the data onto a line
- Red and blue are two classes which can be measured in (x_1, x_2)
- We can take the mean of each class (left), form a line between, then project the data onto the line (middle) giving a distribution (right)
  - We have reduced the 2D data into a 1D projection
- After the projection, the logistic rejection chooses a cut point (via β_0) then sends things below the cut to 0, above to 1
- Here, we see some separation between the classes but a lot of overlap. We can do better

Illustration: Better Fit

- Finding (next page) this discriminant for our illustrative dataset shows that these two classes are fully separable
- The Logistic Regression will place the cut point between the data and so all red go to 0, blue go to 1 after passing through the logistic function

https://medium.freecodecamp.org/an-illustrative-introduction-to-fishers-linear-discriminant-9484efee15ac

Logistic Regression (cont'd)

- In order to implement the minimizer in code, we can take a slightly different approach
- We can write the probability for y given a datapoint x in terms of $f(x, \beta)$
  - If $y = 1$, then $p(y|x, \beta) = f(x, \beta)$
  - If $y = 0$, then $p(y|x, \beta) = 1 - f(x, \beta)$
  - $p(y|x, \beta) = f(x, \beta)^y (1 - f(x, \beta))^{1-y}$ is a combination giving the right function for both
  - Maximizing this probability equivalent to reducing the residuals
- Now, as we saw in stats class, we think of the model as a *likelihood* then to take the log likelihood to minimize
  - $\log L(\beta|x_i, y_i) = y_i \log f(x_i, beta) + (1 - y_i) \log(1 - f(x_i, \beta))$
  - Maximizing the log-likelihood is equivalent to maximizing the likelihood
  - Maximizing the log-likelihood is equivalent to minimizing the negative-log-likelihood
- The minimum negative log-likelihood will be passed to our SGD minimiser from last week to find the best β for logistic regression

Exercises

- Implement `logistic_fn(x)` and `logistic_prime_fn(x)`
  - `logistic_fn` is $f(x) = \frac{1}{1+exp(-x)}$
  - `logistic_prime_fn` is $f(x)(1 - f(x))$
- `logistic(x, beta)` [remember β is similarly defined to last week]
  - Takes a single data point and the known value, and current best β, and passes $\beta \cdot x$ to `logistic_fn`
- `logistic_prime_j(x, beta, j)`
  - Derivative with respect to β_j, $= x_{j-1} f(x, \beta)(1 - f(x, \beta))$ if $j \geq 1$ or $f(x, \beta)(1 - f(x, \beta))$ if $j = 0$
- `logistic_log_likelihood(x_i, y_i, beta)`
  - If y is 1, its log(`logistic` (x, β)), if y is 0 its log(1- `logistic` (x, β))
  - You will need to be careful, can get log(0), so protect, and send a very small number instead
- `logistic_log_likelihood_gradient(x_i, y_i, beta)`
  - $\frac{d \log f}{dx} = \frac{df/dx}{f}$.
  - If y = 0, its - `logistic_prime_j` / (1 - `logistic`) for each j
  - If y = 1, - `logistic_prime_j` / `logistic`

Exercises (cont'd)

- Implement logistic regression using SGD `logistic_regression_sgd(x0, x1, beta0)`
  - `x0`, `x1` are data and the known targets from the two different categories (target 0 for `x0`, 1 for `x1`)
  - `beta0` is your starting point for the logistic function
  - Use `logistic_log_likelihood` and `logistic_log_likelihood_gradient` as `f` and `df` in the SGD minimiser
- Check that your logistic regression function works on some test cases
- Run the logistic regression on the diabetes dataset from kaggle
- Write an `accuracy(data, target, f)` function:
  - `data` is a list of datapoints with known output list `target` (0 or 1)
  - `f` takes in a datapoint, if `f(datapoint)>0.5`, the function is correct if the target is 1, else incorrect and vice-versa for `<.5`
  - `accuracy` returns the fraction of datapoints that `f` got correct
- Write out the best beta and accuracy of your logistic regression fit in a file `results.txt`, commit the code and file and push to github

Diabetes Dataset Information

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.
The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Acknowledgements

Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261–265). IEEE Computer Society Press.