Logistic Regression

Reading

- Chapters 4.1, 4.2, 4.4
  - Hastie, Trevor, et al. The elements of statistical learning: data mining, inference, and prediction. Vol. 2. New York: springer, 2009.
  - Available online: https://hastie.su.domains/Papers/ESLII.pdf
- Chapters 4.1, 4.2, 4.3
  - James, Gareth, et al. An introduction to statistical learning. Vol. 112. New York: springer, 2013.
  - Available online: https://www.statlearning.com/

- Logistic regression from a statistical point of view

Overview

Rensselaer

- Similar to linear regression, logistic regression is one of the most established methods in ML/stats
- Logistic regression is usually used in classification settings
  - Word "regression" is used since we're estimating the probabilities of each label given the features
  - The labels are now discrete values (e.g., objects in an image, the presence/absence of a disease)
- One could also extend regression methods for classification (e.g., by thresholding the output of the function f)
  - But those do not typically estimate probabilities
- Logistic regression is an example of a very simple neural network

Example Classification Tasks

- Many classical ML problems are classification tasks
  - Image classification (i.e., object recognition)
  - Determine whether a patient has cancer from MRI images
  - Determine whether an email is ham or spam
- In the context of autonomous systems and control, many problems can also be mapped to classification tasks
  - Decide which route to a destination to take
  - Decide which action to take (out of a finite number)
  - In general, decision making is one of the main parts of autonomous systems (and it is typically a discrete choice)

Linear Classification Setup

- As before, we are given N labeled IID examples: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
  - where $\boldsymbol{x}_i \in \mathbb{R}^p$
  - Unlike in regression, y_i is a discrete label (e.g., cat, dog)
  - We encode labels with integers, i.e., $y_i \in \{1, \ldots, K\}$
- We assume the examples are sampled from $\mathcal{D}$ and are realizations of random variables $(\boldsymbol{X}, Y) \sim \mathcal{D}$
- The goal of classification is to find an f such that

$$Y = f(\boldsymbol{X})$$

  - Same as in regression, modulo the fact that Y is discrete

Probabilistic View of Classification

- The final goal of classification is a function of the form
$$Y = f(\boldsymbol{X})$$
- An even stronger requirement is to output the probabilities for each label, given an example $\boldsymbol{X}$
  - For K labels, consider the K-dimensional vector $\boldsymbol{Y} \in [0,1]^K$
  - The value of each element Y_i represents
$$\mathbb{P}[Y = i|\boldsymbol{X}]$$
  - That implies $\sum_{i=1}^{K} Y_i = 1$
- Thus, the goal of classification is also to develop a function F
$$\boldsymbol{Y} = F(\boldsymbol{X})$$
- F predicts the probabilities of all labels given an example $\boldsymbol{X}$

Probabilistic View of Classification, cont'd

- Thus, the goal of classification is also to develop a function F

$$\boldsymbol{Y} = F(\boldsymbol{X})$$

- Note that we can build a classifier on top of F
  - How?

$$f(\boldsymbol{X}) = argmax_i\, F(\boldsymbol{X})$$

  - i.e., just take the Y_i with highest probability
  - So computing probabilities of labels is strictly harder than just outputting the most likely label
- Both types of approaches exist
  - Logistic regression takes the latter approach
  - Support vector machines only perform classification

Why not use linear regression for classification?

Rensselaer

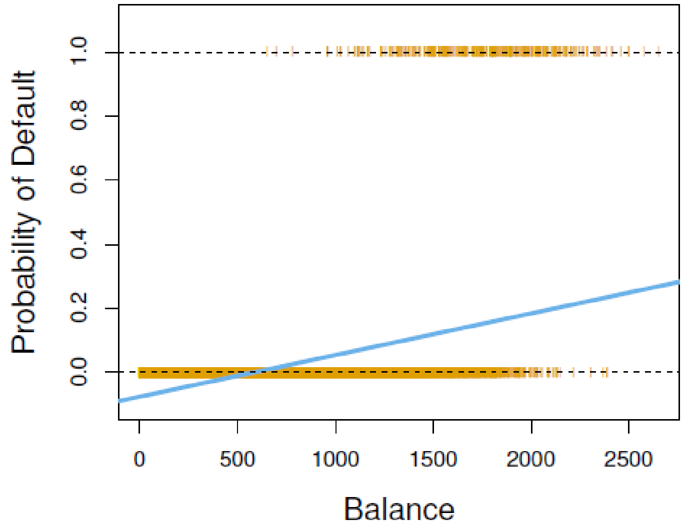
- One could apply regression to classification problems, by using least squares, i.e., minimize

$$\sum_{i=0}^{N}\left(y_i - \boldsymbol{w}^T\boldsymbol{x}_i\right)^2$$

  - where each y_i is an integer
  - Then, predict a discrete label by thresholding $\boldsymbol{w}^T\boldsymbol{x}_i$
    - E.g., in the binary case: $f(\boldsymbol{x}_i) = 1$ if $\boldsymbol{w}^T\boldsymbol{x}_i > 0.5$
- Linear regression is not designed to output probabilities
  - Can output values outside of [0,1]

Linear regression: classification issue in binary case

- Suppose we fit a line and choose a classification threshold
  - Most probabilities for label 1 are very low
  - Some probabilities for label 0 are negative

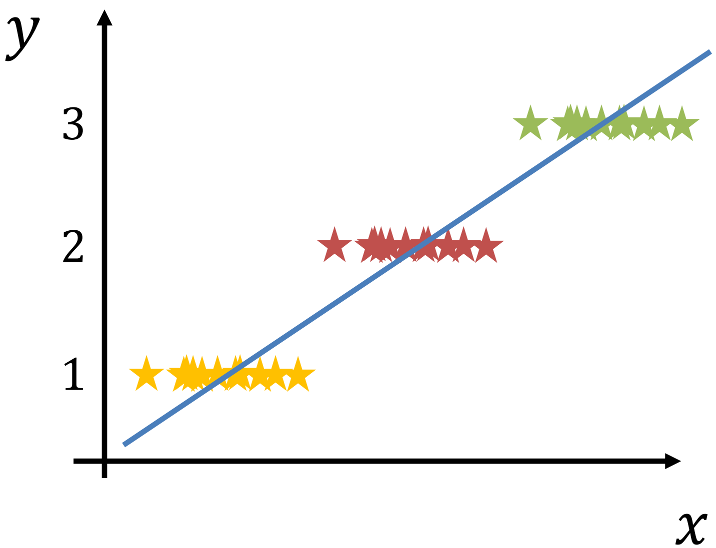


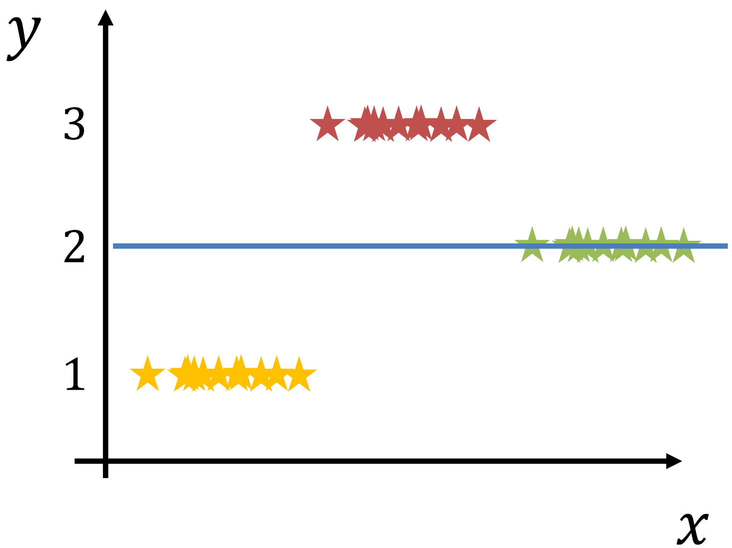
Linear regression: classification issue in multi-label case

- Linear regression gets tricky with multiple labels
- Suppose we are trying to classify an image directly from pixels
  - Labels are: cat, elephant, dog
- What potential issue do you see?
- First of all, assigning number labels to categories is arbitrary
  - E.g., does cat=0, elephant=1, dog=2 make sense?
  - That would imply dog is farther from cat than from elephant
  - We would learn a different function if we change the labels
- Second of all, if we use a linear classifier this way, we would be assuming that a unit difference in y means something

Linear regression: classification issue in multi-label case

- Suppose we have three labels in 1D
  - If we pick the labels right, linear regression works well
  - But if we switch the labels, linear regression loses the middle class
  - How do we address this issue?
    - One option: multiple binary regressions





Logistic Regression

- Linear regression models the labels directly
  - i.e., $Y = f(\boldsymbol{X})$
- Logistic regression models the probability of a given label
  - e.g., in the binary case: $f(\boldsymbol{X}) = \mathbb{P}[Y = 1|\boldsymbol{X}]$
- How do we come up with such a function?
- Can we adapt linear regression to output numbers in [0,1]?
  - Maybe we can normalize the output to be between 0 and 1?
    - Only works if the inputs are bounded
  - Maybe feed the output of linear regression into a function that is always in [0,1]?

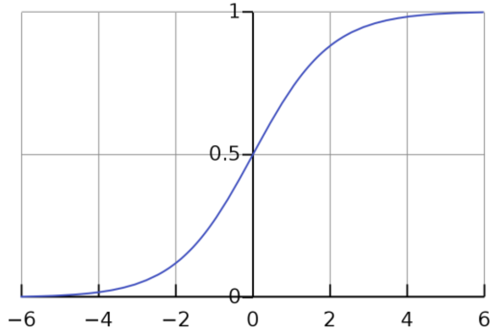
Logistic Regression, cont'd

Rensselaer

- Feed the output of linear regression into a function in [0,1]
  - Solution: the logistic function (also known as the sigmoid)

$$\sigma(x) = \frac{e^x}{1+e^x}$$

    - As $x \to \infty, \sigma(x) \to 1$
    - As $x \to -\infty, \sigma(x) \to 0$



Source: wikipedia

- How do we feed the output of linear regression into σ?

$$f(x) = \frac{e^{w_0 + w_1 x}}{1+e^{w_0 + w_1 x}}$$

- In multiple dimensions (again appending a 1 to $\boldsymbol{x}$):

$$f(\boldsymbol{x}) = \frac{e^{\boldsymbol{w}^T\boldsymbol{x}}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}}}$$

Logistic Regression, cont'd

- In the binary case:

$$\mathbb{P}[Y = 1 | \boldsymbol{X} = \boldsymbol{x}] = \frac{e^{\boldsymbol{w}^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}}$$

  – Similarly, $\mathbb{P}[Y = 0 | \boldsymbol{X} = \boldsymbol{x}] = 1 - \mathbb{P}[Y = 1 | \boldsymbol{X} = \boldsymbol{x}]$

  – i.e.,

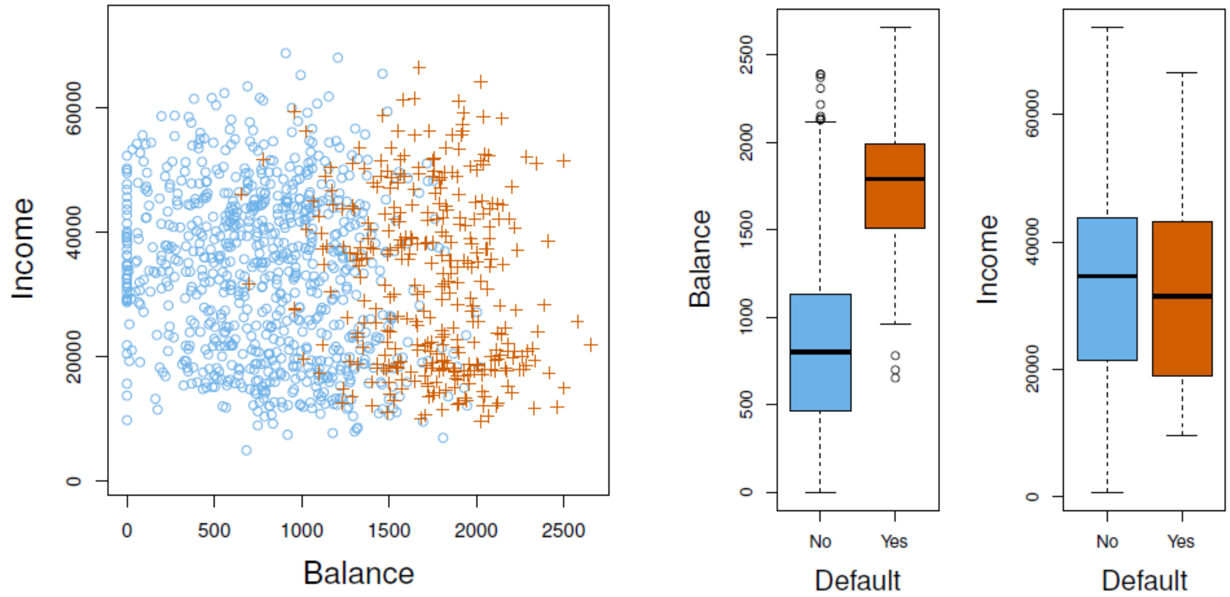
$$\mathbb{P}[Y = 0 | \boldsymbol{X} = \boldsymbol{x}] = 1 - \frac{e^{\boldsymbol{w}^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}}$$

$$= \frac{1 + e^{\boldsymbol{w}^T \boldsymbol{x}} - e^{\boldsymbol{w}^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}}$$

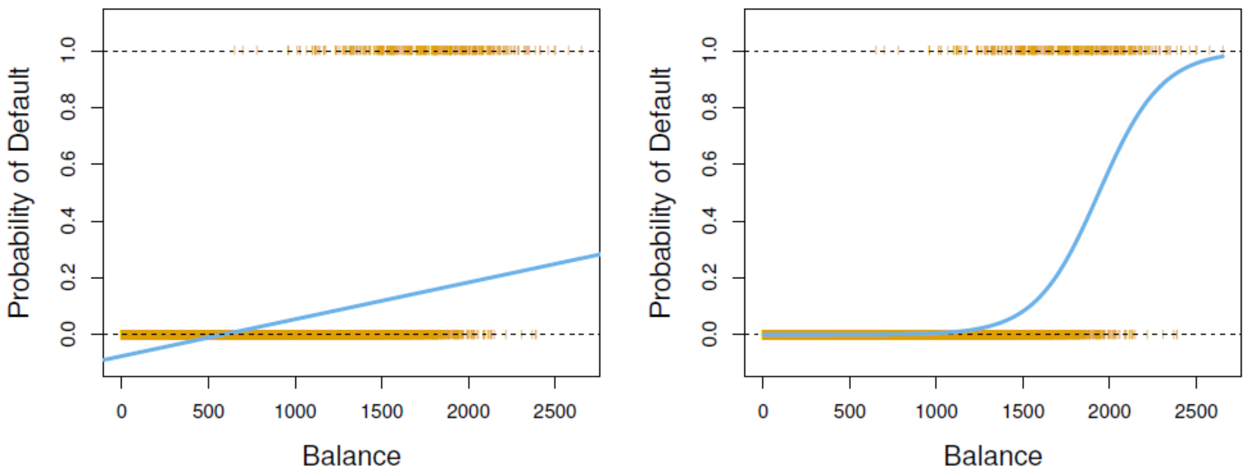
$$= \frac{1}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}}}$$

Example

- Use a simulated dataset from the book
- Goal is to predict whether a person will default on their credit card payment
  - Features are annual income and current balance



Logistic vs. Linear Regression



- Some probabilities predicted by linear regression are negative
- In terms of classification, two methods are the same
  - Why?
  - Classification threshold can be adjusted for each method to maximize classification accuracy

Learning the Logistic Regression Coefficients

- In linear regression, we learned the coefficients using MSE

$$\sum_{i=1}^{N} e_i^2$$

  – where $e_i = y_i - f(\boldsymbol{x}_i)$ are the prediction errors

- We could do the same for logistic regression:

$$\sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left(y_i - \frac{e^{\boldsymbol{w}^T \boldsymbol{x}_i}}{1 + e^{\boldsymbol{w}^T \boldsymbol{x}_i}} \right)^2$$

  – What issues do you see with this expression?
  – It's not quadratic in $\boldsymbol{w}$, so we can't minimize it by hand
  – There exist minimization algorithms, will look at them later in the course

Learning the Logistic Regression Coefficients: Maximum Likelihood Estimation

- An alternative way to learning the coefficients is through maximizing the data likelihood
- The real data is distributed according to an unknown distribution
  - E.g., each example $(\boldsymbol{x}, y)$ has an unknown conditional distribution

$$\mathbb{P}[Y = y|\boldsymbol{X} = \boldsymbol{x}]$$

- For given logistic weights $\boldsymbol{w}$, logistic regression predicts probability (e.g., for $y = 1$)

$$\mathbb{P}_{\boldsymbol{w}}[Y = 1|\boldsymbol{X} = \boldsymbol{x}] = \frac{e^{\boldsymbol{w}^T\boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T\boldsymbol{x}}}$$

  - Pick weights $\boldsymbol{w}$ that maximize predicted training data probability

Learning the Logistic Regression Coefficients: Maximum Likelihood Estimation, cont'd

- True data likelihood can be simplified

$$\mathbb{P}[y_1, \ldots, y_N | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] = \\ = \prod_{i=1}^{N} \mathbb{P}[y_i | \boldsymbol{x}_i]$$

- Why?
  – Data is IID
    - Joint probability is equal to the product of individual probabilities
- How do we maximize the predicted likelihood by the sigmoid?
  – Choose weights $\boldsymbol{w}$ that maximize predicted likelihood

$$\prod_{i=1}^{N} \mathbb{P}_{\boldsymbol{w}}[y_i | \boldsymbol{x}_i]$$

Learning the Logistic Regression Coefficients: Maximum Likelihood Estimation, cont'd

Rensselaer

- Instead of maximizing the likelihood, we are actually going to maximize the logarithm of likelihood

$$LL = \log\left(\prod_{i=1}^{N} \mathbb{P}_{\boldsymbol{w}}[y_i|\boldsymbol{x}_i]\right)$$

- Claim: the $\boldsymbol{w}$ that maximizes the likelihood also maximizes the log-likelihood (why?)
  - Logarithm is monotonic
  - So maximizing the log-likelihood is the same as maximizing the likelihood

$$LL = \log\left(\prod_{i=1}^{N} \mathbb{P}_{\boldsymbol{w}}[y_i|\boldsymbol{x}_i]\right) = \sum_{i=1}^{N} \log(\mathbb{P}_{\boldsymbol{w}}[y_i|\boldsymbol{x}_i])$$

Learning the Logistic Regression Coefficients: Maximum Likelihood Estimation, cont'd

$$LL = \sum_{i=1}^{N} \log(\mathbb{P}_{\boldsymbol{w}}[y_i|\boldsymbol{x}_i])$$

- Note $\mathbb{P}_{\boldsymbol{w}}[y_i = 1|\boldsymbol{x}_i] = \frac{e^{\boldsymbol{w}^T\boldsymbol{x}_i}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}$ and $\mathbb{P}_{\boldsymbol{w}}[y_i = 0|\boldsymbol{x}_i] = \frac{1}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}$
- So we can write

$$\log(\mathbb{P}_{\boldsymbol{w}}[y_i|\boldsymbol{x}_i]) = y_i \log\left(\frac{e^{\boldsymbol{w}^T\boldsymbol{x}_i}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}\right) + (1-y_i)\log\left(\frac{1}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}\right)$$

$$= y_i \log\left(\frac{e^{\boldsymbol{w}^T\boldsymbol{x}_i}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}} \frac{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}{1}\right) + \log\left(\frac{1}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}\right)$$

$$= y_i \log\left(e^{\boldsymbol{w}^T\boldsymbol{x}_i}\right) + \log\left(\frac{1}{1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}}\right)$$

Learning the Logistic Regression Coefficients: Maximum Likelihood Estimation, cont'd

Rensselaer

$$LL = \sum_{i=1}^{N} y_i \boldsymbol{w}^T \boldsymbol{x}_i - \log\left(1 + e^{\boldsymbol{w}^T \boldsymbol{x}_i}\right)$$

- To find the maximizing $\boldsymbol{w}$, take the derivative w.r.t. $\boldsymbol{w}$ and set it equal to 0
  - Logistic regression LL is a concave function in $\boldsymbol{w}$
- Unfortunately, the derivative becomes a transcendental equation, so it has no closed-form solution ☺
  - Similar to non-linear least squares, algorithms exist for solving this numerically
    - We'll look at them later in the course

Loss functions

$$LL = \sum_{i=1}^{N} y_i \boldsymbol{w}^T \boldsymbol{x}_i - \log\left(1 + e^{\boldsymbol{w}^T \boldsymbol{x}_i}\right)$$

- ML people like to minimize functions (instead of maximize), so we typically minimize the negative log-likelihood:

$$NLL = -\left(\sum_{i=1}^{N} y_i \boldsymbol{w}^T \boldsymbol{x}_i - \log\left(1 + e^{\boldsymbol{w}^T \boldsymbol{x}_i}\right)\right)$$

- Negative log-likelihood and least squares are our first examples of loss functions
  - More later

Multinomial Logistic Regression

- What about the case of multiple labels?
  - All probabilities must sum up to 1

$$\mathbb{P}_{\boldsymbol{w}}[Y = 1 | \boldsymbol{X} = \boldsymbol{x}] + \cdots + \mathbb{P}_{\boldsymbol{w}}[Y = K | \boldsymbol{X} = \boldsymbol{x}] = 1$$

- We need a separate weight vector for each label

$$f_i(\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_i^T \boldsymbol{x}}}{1 + e^{\boldsymbol{w}_i^T \boldsymbol{x}}}$$

- Then normalize

$$\mathbb{P}_{\boldsymbol{w}}[Y = i | \boldsymbol{X} = \boldsymbol{x}] = \frac{f_i(\boldsymbol{x})}{\sum_{i=1}^{K} f_i(\boldsymbol{x})}$$

- This approach is called multinomial logistic regression
  - Also known as softmax in deep learning

Multinomial Logistic Regression, cont'd

- Probability for each label is

$$\mathbb{P}_{\boldsymbol{w}}[Y = i | \boldsymbol{X} = \boldsymbol{x}] = \frac{f_i(\boldsymbol{x})}{\sum_{i=1}^{K} f_i(\boldsymbol{x})}$$

- Now, LL becomes

$$LL = \sum_{i=1}^{N} \log(\mathbb{P}_{\boldsymbol{w}}[y_i | \boldsymbol{x}_i])$$
$$= \sum_{i=1}^{N} \log\left(\frac{f_{y_i}(\boldsymbol{x}_i)}{\sum_{j=1}^{K} f_j(\boldsymbol{x}_i)}\right)$$

- Maximizing the LL is once again done using specialized algorithms based on gradient descent