Tufts

Class #07: Logistic Regression

Machine Learning (COMP 135): M. Allen, 25 Sept. 19

Reminder: Threshold Functions

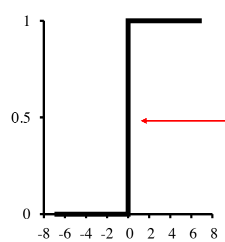
1. We have data-points with n features:
$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$
2. We have a linear function defined by $n+1$ weights:
$$\mathbf{w} = (w_0, w_1, w_2, \ldots, w_n)$$
3. We can write this linear function as:
$$\mathbf{w} \cdot \mathbf{x}$$
4. We can then find the linear boundary, where:
$$\mathbf{w} \cdot \mathbf{x} = 0$$
5. And use it to define our threshold between classes:
$$h_{\mathbf{w}} = \begin{cases} 1 & \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \mathbf{w} \cdot \mathbf{x} < 0 \end{cases}$$

Outputs 1 and 0 here are ***arbitrary labels*** for one of two possible classes

Hard Thresholds are Hard!

$$h_{\mathbf{w}} = \begin{cases} 1 & \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \mathbf{w} \cdot \mathbf{x} < 0 \end{cases}$$

- The hard threshold function used by the perceptron algorithm (among others) produces some conceptual and mathematical challenges
- Gives a yes/no answer everywhere, which can be tricky when our data isn't linearly separable

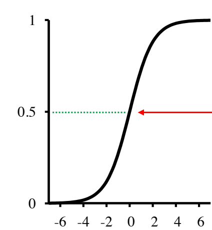


Function is discontinuous (non-differentiable) at $x = 0$

The Logistic Function

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

- We can generate a smooth curve by instead using the logistic function as a threshold
- We can treat this value as a ***probability*** of belonging to one class or another

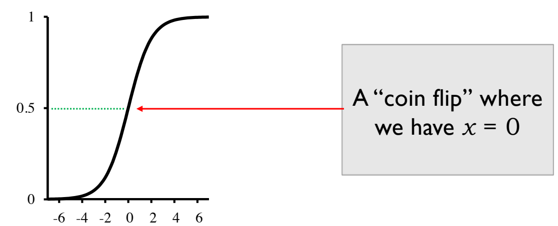


Probability function is 0.5 at $x = 0$

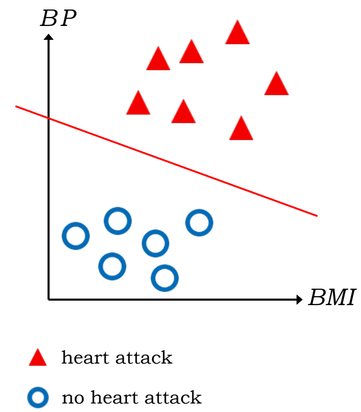
Using the Logistic for Classification

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

- Treated as a probability, the logistic can still be used to *classify* data, where the class is the one that has highest probability overall, while also supplying a probability for that outcome

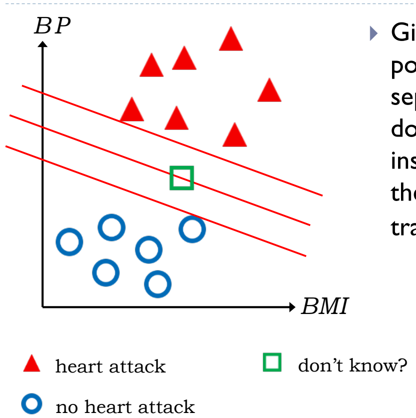


Issues with Linear Classification



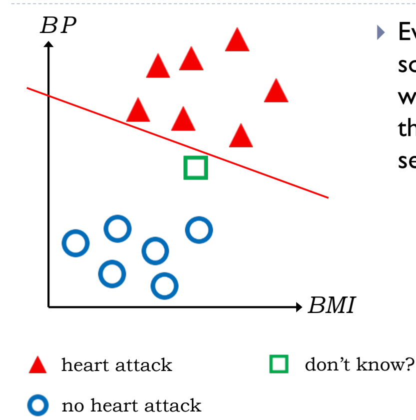
- Consider data about heart-attack risk, based upon body mass index (BMI) and blood pressure (BP)
- Even assuming linearly separable training data, linear classification gives a hard cut-off that may not be appropriate

Issues with Linear Classification



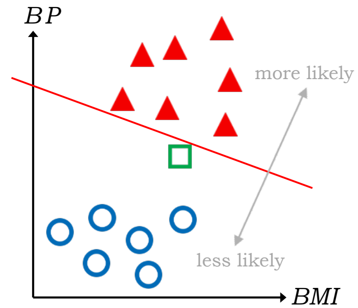
- Given that *multiple* possible lines can separate this data, how do we classify a new instance when it lies in the region between the training instances?

Issues with Linear Classification



- Even if we did settle on some fixed line, what do we do with something that is *very close* to the separator?

Using Probabilistic Classification

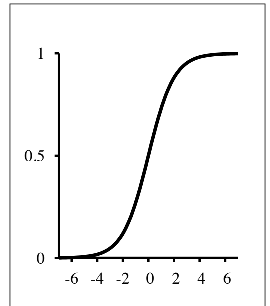


▲ heart attack □ don't know?

○ no heart attack

- Logistic regression also generates a linear separator (where the weight-function = 0), but now it is giving us a distribution over data
- A new data point close to the line still has *some positive probability* of being in the class on the other side of it

Properties of the Logistic Function



$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}\cdot\mathbf{x}}}$$

- Also known as the Sigmoid, from the shape of its plot
- It always has a value in range:

  $0 \leq x \leq 1$
- The function is *everywhere* differentiable, and has a *derivative* that is easy to calculate, which turns out to be useful for learning:

$$h'_{\mathbf{w}}(\mathbf{x}) = h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x}))$$

Logistic Regression

- In perceptron learning we update the weight vector in each case based upon a mis-classified instance, using the equation:

$$w_j \leftarrow w_j + \alpha(y_i - h_{\mathbf{w}}(\mathbf{x}_i)) \times x_{i,j}$$

- In the case of the logistic, we do the same, but add an extra term:

$$w_j \leftarrow w_j + \alpha(y_i - h_{\mathbf{w}}(\mathbf{x}_i)) \times h_{\mathbf{w}}(\mathbf{x}_i)(1 - h_{\mathbf{w}}(\mathbf{x}_i)) \times x_{i,j}$$

The difference between what output ***should*** be, and what our weights make it

The *j*th feature-value

The derivative of the logistic

Applying the Logistic

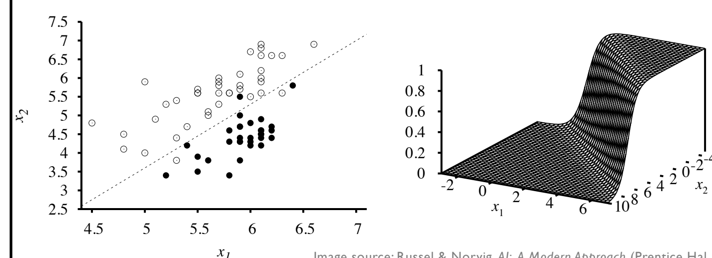


Image source: Russel & Norvig, *AI: A Modern Approach* (Prentice Hal, 2010)

- When we have data that is not linearly separable, our hard threshold still has to make a hard decision
- With the logistic, we get a smooth surface where things close to the boundary between classes are only *probably* in one or the other

Gradient Descent for Logistic Regression

- We can use the same approach as for linear classification, starting with some random (or uniform) weights and then:

1. Choose an input $\mathbf{x}_i$ from our data set that is wrongly classified.
2. Update vector of weights, $\mathbf{w} = (w_0, w_1, w_2, \ldots, w_n)$:

$$w_j \leftarrow w_j + \alpha(y_i - h_{\mathbf{w}}(\mathbf{x}_i)) \times h_{\mathbf{w}}(\mathbf{x}_i)(1 - h_{\mathbf{w}}(\mathbf{x}_i)) \times x_{i,j}$$

3. Repeat until weights no longer change; modify learning parameter α over time to guarantee this.

- Again, we make α smaller and smaller over time, and the algorithm converges as $\alpha \to 0$

Gradient Descent for Logistic Regression

$$w_j \leftarrow w_j + \alpha(y_i - h_{\mathbf{w}}(\mathbf{x}_i)) \times h_{\mathbf{w}}(\mathbf{x}_i)(1 - h_{\mathbf{w}}(\mathbf{x}_i)) \times x_{i,j}$$

- The logistic update equation, via gradient descent, minimizes the log loss (as seen in last lecture), also known as the binary cross entropy:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1 - y_i)\log(1 - p_i)$$

  - For these purposes, we treat the output of the logistic as the probability we are interested in: $p_i \triangleq h_{\mathbf{w}}(\mathbf{x}_i)$
  - Over time, we drive the loss towards 0

Logarithmic Loss vs. Error

- For an individual data element, the log loss is an upper bound on the basic (1/0) loss previously considered:

$$\mathcal{E}(h_{\mathbf{w}}(\mathbf{x}_i), y_i) = \begin{cases} 0 & \text{if } h_{\mathbf{w}}(\mathbf{x}_i) = y_i \\ 1 & \text{if } h_{\mathbf{w}}(\mathbf{x}_i) \neq y_i \end{cases}$$

$$\mathcal{L}(h_{\mathbf{w}}(\mathbf{x}_i), y_i) = -[y_i \log h_{\mathbf{w}}(\mathbf{x}_i) + (1 - y)\log(1 - h_{\mathbf{w}}(\mathbf{x}_i))]$$

- This graph assumes:
  1. True label is 1
  2. Threshold used is 0.5
  3. Log base 2 is used

Linear vs. Logistic Regression for Classification Purposes

Linear Regression	Logistic Regression
A value x ∈ ℝ	A value 0 ≤ x ≤ 1
A hard boundary between classes on either side of a line	Probability of belonging to a certain class
Tries to find line that best ***fits*** the data	Tries to find separator that best ***divides*** the classes

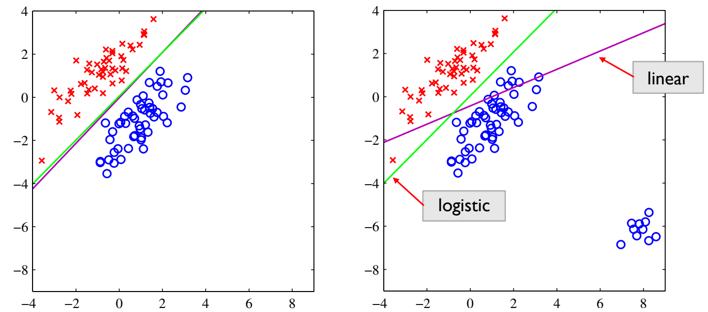
Linear vs. Logistic Regression in Mathematical Terms

Linear	
Loss function	$Loss(\mathbf{w}) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2$
Weight-update equation	$w_i \leftarrow w_i + \alpha \sum_j x_{j,i} \, (y_j - h_{\mathbf{w}}(\mathbf{x}_j))$

Logistic	
Loss function	$-\frac{1}{N} \sum_{j=1}^{N} [y_j \log h_{\mathbf{w}}(\mathbf{x}_j) + (1 - y_j) \log(1 - h_{\mathbf{w}}(\mathbf{x}_j))]$
Weight-update equation	$w_j \leftarrow w_j + \alpha (y_j - h_{\mathbf{w}}(\mathbf{x}_j)) \times h_{\mathbf{w}}(\mathbf{x}_j)(1 - h_{\mathbf{w}}(\mathbf{x}_j)) \times x_{i,j}$

Treatment of Outliers in Data

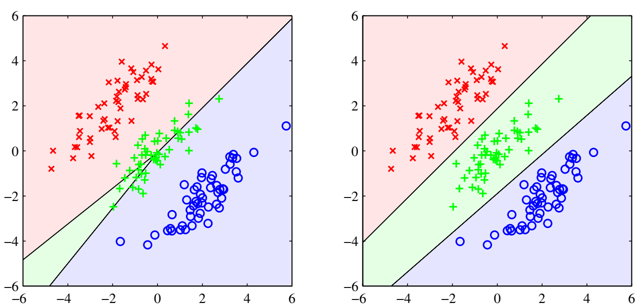
Images from: C. Bishop, *Pattern Recognition and Machine Learning*. Springer (2006).



- Logistic regression (green) and linear regression (magenta) give similar results on some data
- Linear regression is skewed by outliers, however, as loss function sees them as "too correct"

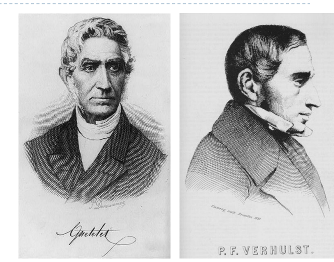
Classifier Performance

Images from: C. Bishop, *Pattern Recognition and Machine Learning*. Springer (2006).



- Linear regression has trouble separating data in some cases
- The green data are almost all incorrect for this 2-line regression
- Logistic regression (again with 2 distinct lines of separation, using 2 different regressions) performs well on same data

History of Logistic Regression (1838–1847)



- The logistic function and its name come from three papers by Pierre François Verhulst (right), a statistician and student of Alphonse Quételet (left)
- They were interested in modeling human population growth, which will tend to grow exponentially unless checked, but has an upper bound (equilibrium) at which it maxes out and stops growing
- The Sigmoid curve was a good fit for real population data for France, Belgium, and Russia up to the year 1833

History of Logistic Regression (20th C.)

- The logistic was re-discovered by Raymond Pearl (left) and Lowell Reed (right) in the 1920's
- They later discovered Verhulst's earlier work, and credited him, but his logistic terminology didn't really catch on until the work of others, after WWII
- Pearl and collaborators went on to apply the logistic curve to models of human and fruit fly populations, as well as to the growth of cantaloupes

- In the 40's and 50's, statisticians working to model bioassay (effects of medicines and other substances on living tissues) popularized the use of the logistic and its name
- Due to computational conveniences, this became more popular than other models

This Week & Next

- Logistic Regression and Nearest Neighbors Clustering
- Readings:
  - Linked from class website schedule page
  - Information on the history of logistic regression can be found in J. S. Cramer, "The Origins of Logistic Regression," Tinbergen Institute, 2002.
- Homework 02: due Wednesday, 02 October, 9:00 AM
  - Get Python environment up and running soon!
- Office Hours: 237 Halligan, Tuesday, 11:00 AM – 1:00 PM
  - TA hours can be found on class website as well