Logistic Regression

Dr. Anahita Zarei

Why Logistic Regression?

- There are many applications where we are not only interested in the predicted class labels, but where the estimation of the class-membership probability is particularly useful (the output of the sigmoid function prior to applying the threshold function).
- Logistic regression is used in weather forecasting, for example, not only to predict if it will rain on a particular day but also to report the chance of rain.
- Similarly, logistic regression can be used to predict the chance that a patient has a particular disease given certain symptoms, which is why logistic regression enjoys great popularity in the field of medicine.

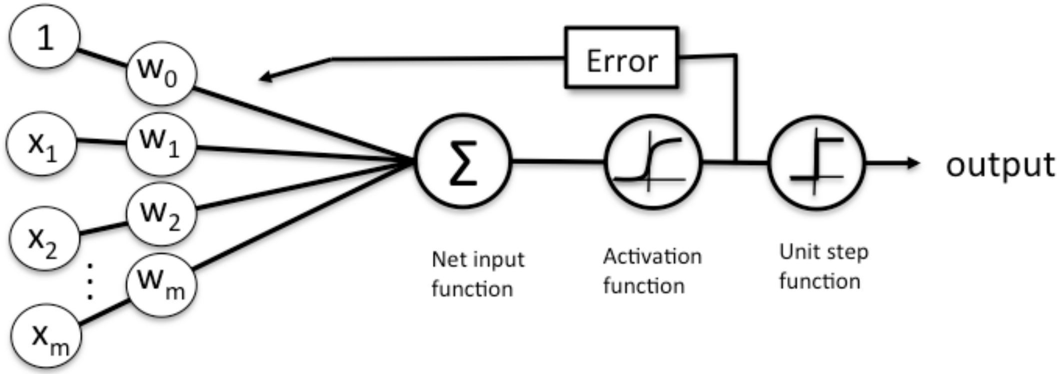


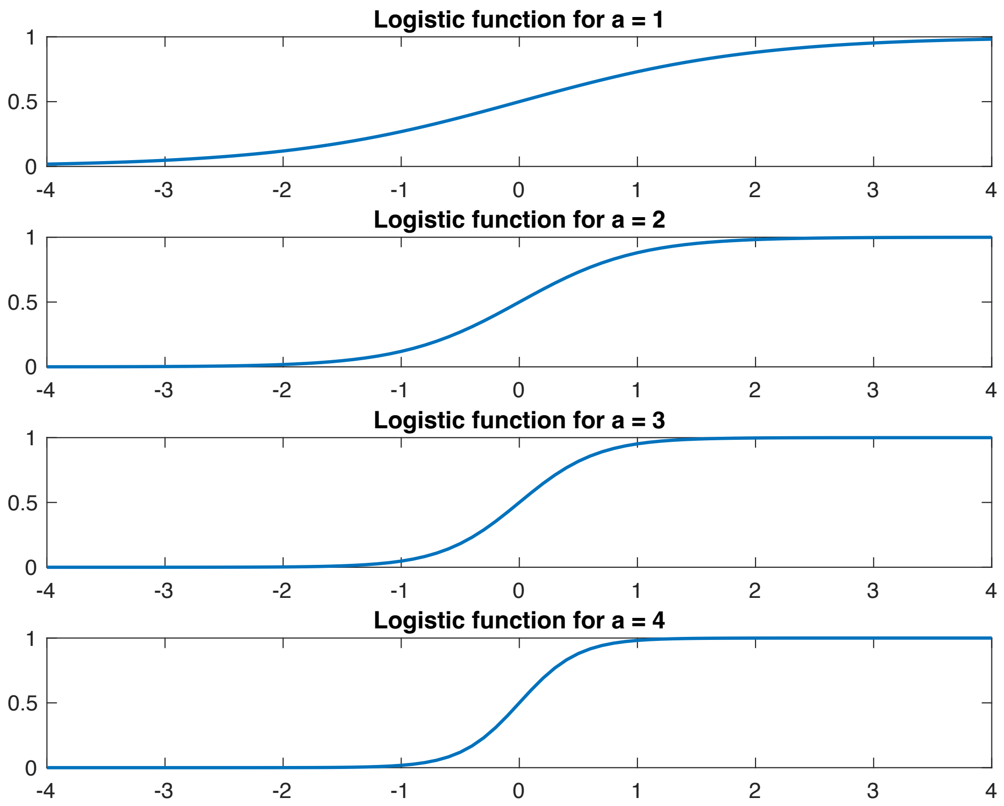
Image source:
http://rasbt.github.io/mlxtend/user_guide/classifier/LogisticRegression/

Example: Prediction of Heart attacks

- Predict the occurrence of heart attacks based on a person’s:
  - cholesterol level (LDL),
  - blood pressure,
  - age,
  - weight.
- We cannot predict heart attack with certainty, but we can predict the probability.
- Therefore, the output will be a continuous function between 0 and 1.
- The closer y (the output of the model) to 1, the more likely that the person will have a heart attack.

Logistic Function

- Logistic Function smoothly restricts the output to the probability range [0,1].
- The equation is given by $\mathrm{f}(x) = \frac{1}{1+e^{-ax}} = \frac{e^{ax}}{1+e^{ax}}$
- This specific formula of f(x) will allow us to define an error measure for learning that has analytical and computational advantages.

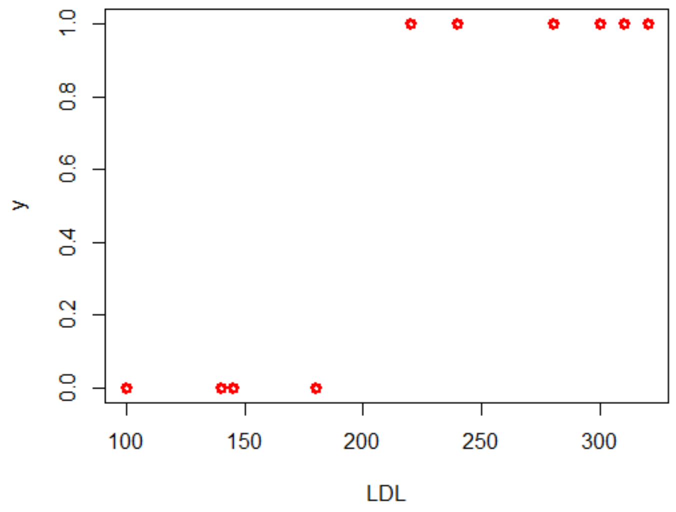


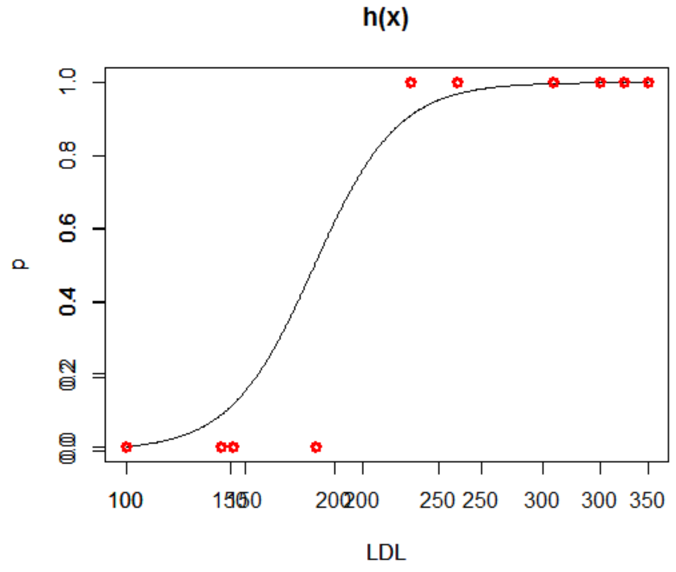
Hypothesis Function

- Let's assume that we would like to predict the probability of heart attack based on a single attribute, namely LDL.
- Our training set is denoted by red dots.
- The goal is to learn the hypothesis function $h(x) = P[y = 1|x]$
- The data doesn't give us the value of h(x) explicitly, but it gives us samples generated by this probability function.
- Therefore, the data is generated by a noisy target :

$$\mathrm{P}[y|x] = \begin{cases} h(x) & y = 1 \\ 1 - h(x) & y = 0 \end{cases}$$

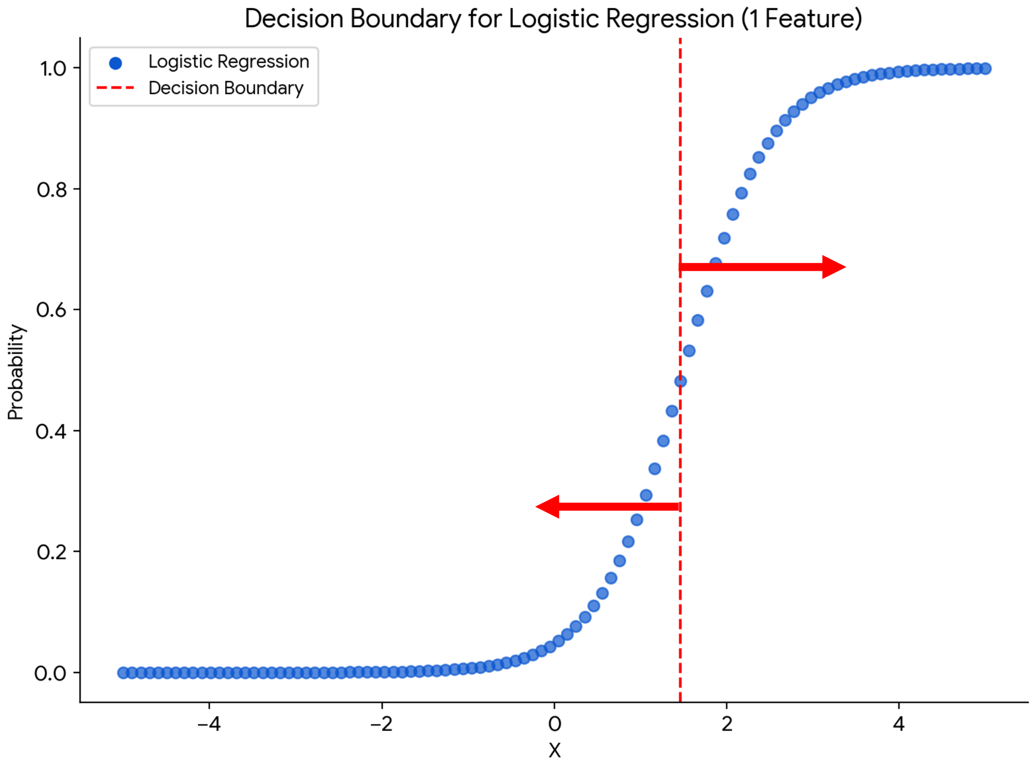
- Based on the training information we must find the parameters of h(x).





Decision Boundary

- Predict y = 1 if P[y|X] > 0.5 ⇒ $\frac{1}{1+e^{-\sum w_i x_i}} > 0.5$
- Predict y = 0 if P[y|X] < 0.5⇒ $\frac{1}{1+e^{-\sum w_i x_i}} < 0.5$
- $\frac{1}{1+e^{-\sum w_i x_i}} = 0.5$ (0.5 is threshold for making decision)
- $\frac{1}{1+e^{-\sum w_i x_i}} = 0.5 \Rightarrow 1 + e^{-\sum w_i x_i} = 2 \Rightarrow$ $e^{-\sum w_i x_i} = 1 \Rightarrow -\sum w_i x_i = log1 \Rightarrow$
- $\sum w_i x_i = 0$
- $\sum w_i x_i > 0 \Rightarrow$ Predict 1
- $\sum w_i x_i < 0 \Rightarrow$ Predict 0
- Anything to the right of the vertical line (in this case w0+w1x1>0) will map to 1 (i.e. p > 0.5)
- Anything to the left of the vertical line (in this case w0+w1x1<0) will map to 0 (i.e. p < 0.5)

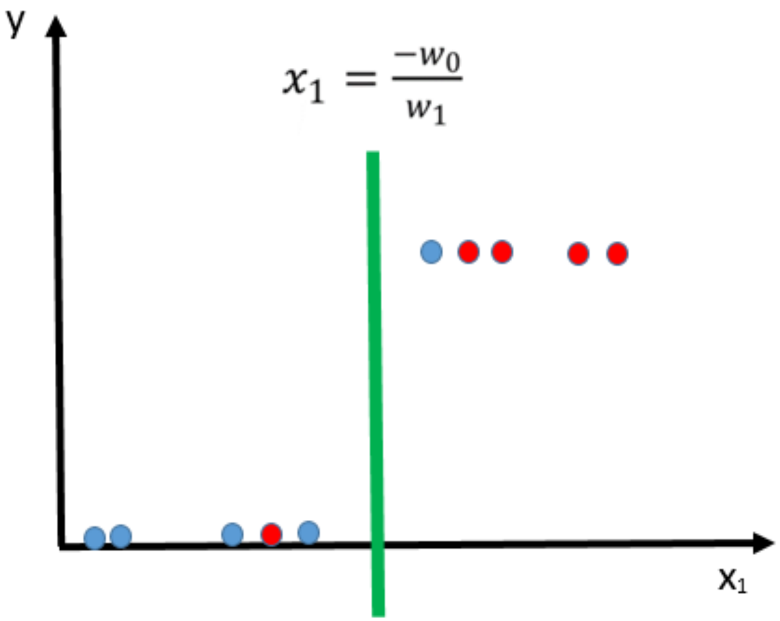


The x-axis represents the values of the feature, and the y-axis represents the predicted probability. The sigmoid curve shows the output of the logistic regression model, which is a probability between 0 and 1. The vertical dashed line (X = -w0/w1) is the decision boundary, which separates the two classes.

Decision Boundary

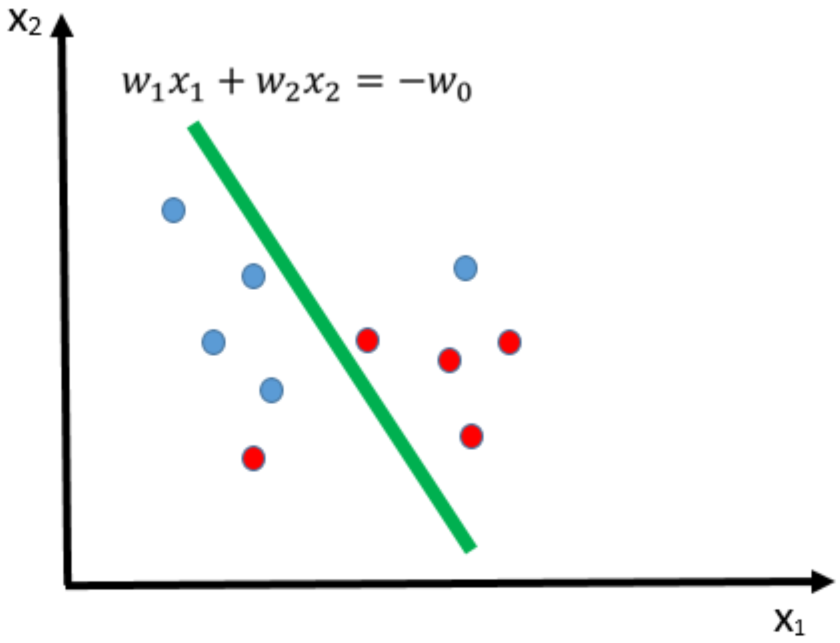
Example: one feature

- $w_0 + w_1 x_1 = 0 \Rightarrow x_1 = \frac{-w_0}{w_1}$



Example: two features

- $w_0 + w_1 x_1 + w_2 x_2 = 0 \Rightarrow$
- $w_1 x_1 + w_2 x_2 = -w_0$



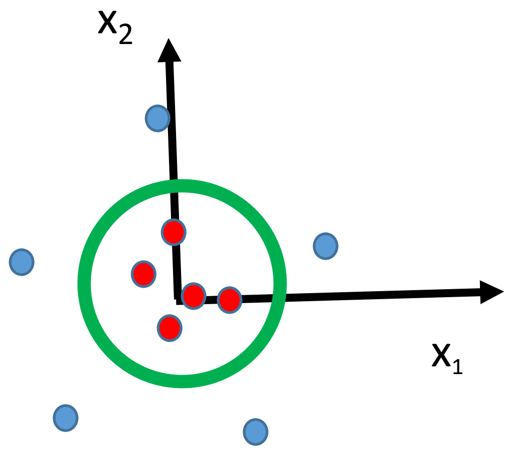
Non-linear Decision Boundary

- Predict y = 1 if $\frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2+w_3x_1^2+w_4x_2^2)}} > 0.5$
- Predict y = 0 if $\frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2+w_3x_1^2+w_4x_2^2)}} < 0.5$
- $\frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2+w_3x_1^2+w_4x_2^2)}} = 0.5$ is the decision boundary.
- $\frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2+w_3x_1^2+w_4x_2^2)}} = 0.5 \Rightarrow$

- $w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 = 0$

Non-linear Decision Boundary-Example

- $h(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$
- Let's say $w = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$
- Predict y = 1 if $-1 + x_1^2 + x_2^2 \geq 0$
- Predict y = 0 if $-1 + x_1^2 + x_2^2 < 0$



Representing Data

- $input = \vec{X} = [x_1 \ldots x_d]$
- For example in the heart attack problem
  $$\textbf{input} = [\boldsymbol{x_1 = LDL, x_2 = blood\ pressure, x_3 = age, x_4 = weight}]$$
- Assume we have this data available for N patients. We will use superscript to denote the number of subjects, and subscripts to denote the number of attributes.
- We can denote the input in a matrix form as follows. Each row corresponds to information about a given patient. Ideally this would be a tall matrix (i.e. much more patients than features): $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x_1^1} & \cdots & \boldsymbol{x_d^1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x_1^N} & \cdots & \boldsymbol{x_d^N} \end{bmatrix}$
- We will denote the output by y. We assume that y is binary (0 or 1). We will denote the output in a column vector as follows:
- $\vec{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y^1} \\ \vdots \\ \boldsymbol{y^N} \end{bmatrix}$

Representing the Hypothesis

- $h(\vec{x}) = \frac{1}{1+e^{-(w_0+w_1x_1+\cdots+w_dx_d)}}$

- Let $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$ and $\vec{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$ then you can write the hypothesis as

$$h(\vec{x}) = \frac{1}{1 + e^{-\vec{w}^T\vec{x}}} = \frac{1}{1 + e^{-\sum w_i x_i}}$$

Example: Prediction of Heart attacks

- Assume your input is $\vec{x} = \begin{bmatrix} 1 \\ LDL = 270 \\ Blood\ Pressure = 130 \\ Age = 60 \\ Weight = 200 \end{bmatrix}$.
- Suppose your hypothesis returns $h(\vec{x})$= 0.75.
- This means that a patient with above attribute has a probability of 0.75 of getting a heart attack.
- In other words $\mathrm{p}(\mathrm{y} = 1|\vec{x}) = 0.75$ or equivalently

$$\mathrm{p}(\mathrm{y} = 0|\vec{x}) = 1 - h(\vec{x}) = 0.25$$

How to Find the Model Parameters: Maximum Likelihood Criteria

- Logistic Regression assumes a parametric form for the distribution $P(y|\vec{x})$.
- One reasonable approach to training Logistic Regression is to choose parameter values that maximize the conditional data likelihood.
- The conditional data likelihood is the probability of the observed y values in the training data, conditioned on their corresponding $\vec{x}$ values.
- We choose parameters that satisfy

$$\vec{w} = argmax \prod_{i=1}^{N} p(y^i|\vec{x}^i;\vec{w})$$

Where $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$ and y^i denote the observed value of y in the i^{th} training example, and , $\vec{x^i} = \begin{bmatrix} 1 \\ x_1^i \\ \vdots \\ x_d^i \end{bmatrix}$ denotes the observed value of x in the i^{th} training example. **How likely is it to see the actual outcomes (y) that we observed, given the input features (X) and the model's weights (w).**

- The expression to the right of the argmax is the conditional data likelihood $l(\vec{w})$.
- Here we include vector w in the conditional, to emphasize that the expression is a function of the w we are attempting to maximize.

Maximum Likelihood Expression

- Equivalently, we can work with the log of the conditional likelihood.

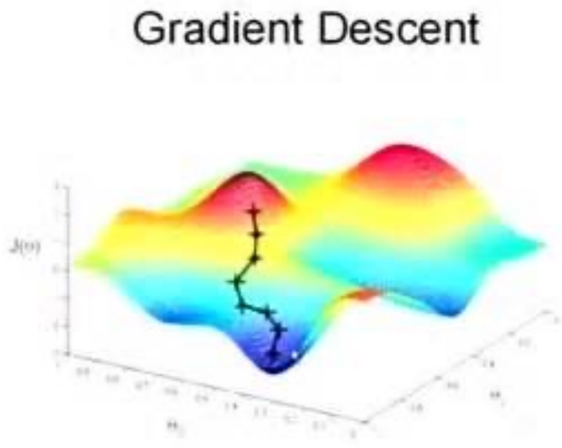
$$\log l(\vec{w}) = \log \prod_{i=1}^{N} p(y^i|\vec{x}^i;\vec{w}) = \sum_{i=1}^{N} \log p(y^i|\vec{x}^i;\vec{w})$$

- Recall $\log \mathrm{P}[y|\vec{x}] = \begin{cases} \log h(\vec{x}) & y = 1 \\ \log(1 - h(\vec{x})) & y = 0 \end{cases}$
- We can combine the above two statements in a single statement as follows:
  $\log P[y|\vec{x}] = y \log h(\vec{x}) + (1 - y) \log(1 - h(\vec{x}))$
- Combining the red and blue expression above will result in:

$$\log l(\vec{w}) = \sum_{i=1}^{N} y^i \log h(\vec{x}^i) + \left(1 - y^i\right) \log(1 - h(\vec{x}^i))$$

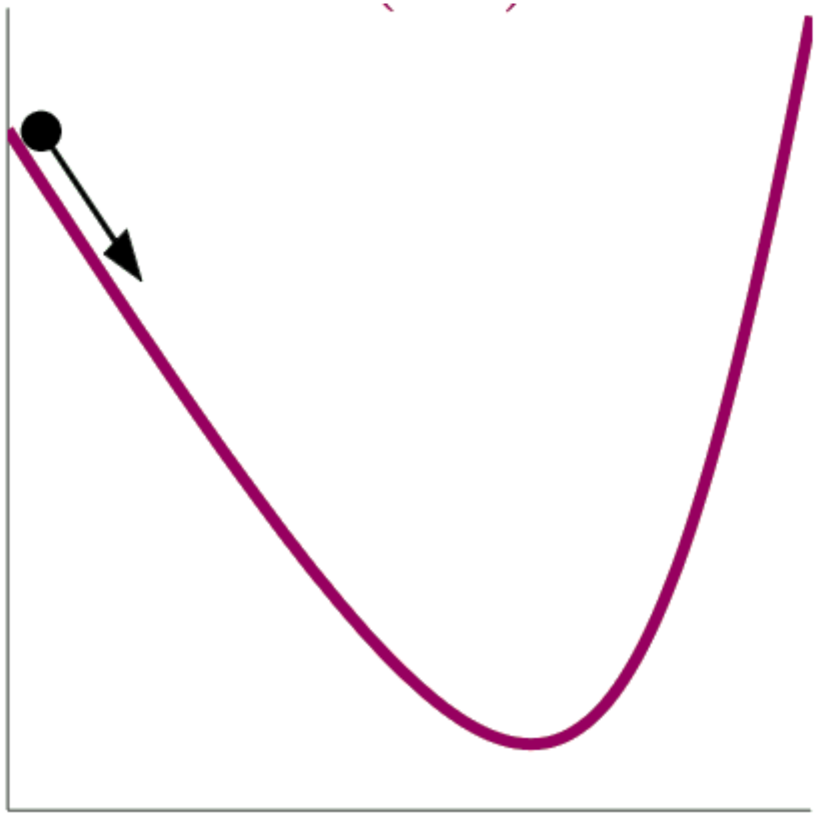
Gradient Descent/Ascent

- Gradient Descent is a general technique for minimizing/maximizing a function.
- You can think of the algorithm as a ball rolling down a hilly surface.
- It the ball is placed on a hill, it will roll down, coming to rest at the bottom of a valley.



Convex Function

- If you have a convex function, there is only one valley.
- Therefore the ball will always roll down the same global minimum.
- This implies that gradient descent will not be trapped in local minima when minimizing such convex function.



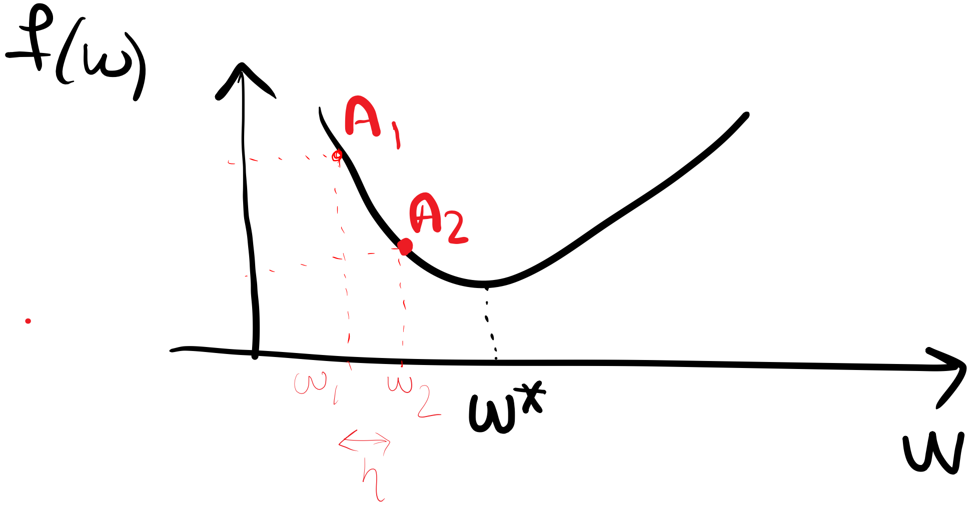
Best Direction to Climb/Descent

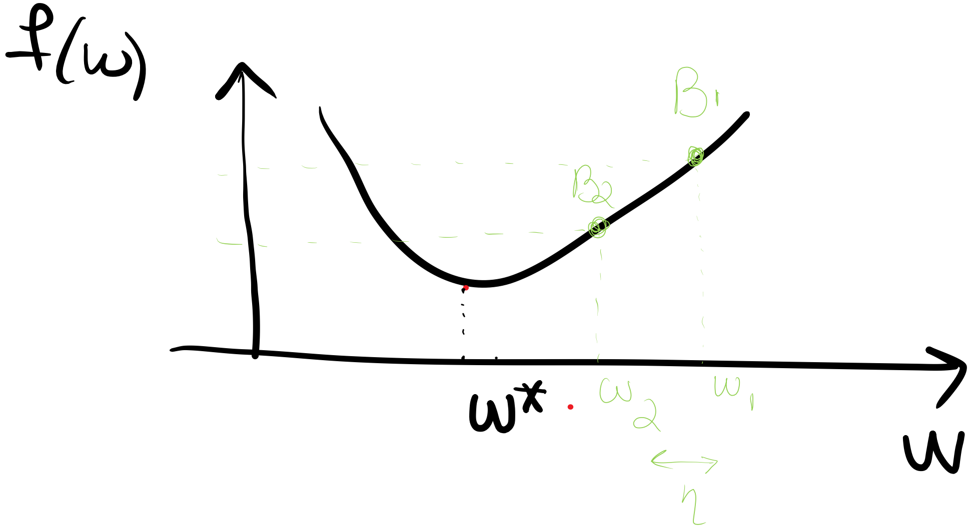
The Rule for Updating w

- $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$

- $new\ w = old\ w \ - \ \eta \frac{\nabla f(w)}{\|\nabla f(w)\|}$

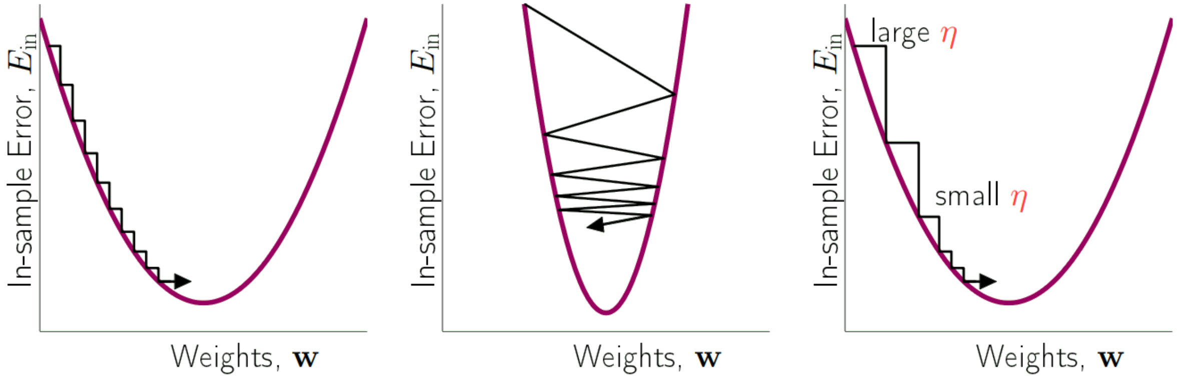
- $w_2 = w_1 \ - \ \eta \frac{\nabla f(w)}{\|\nabla f(w)\|}$





Step Size

- $new\ w = old\ w\ - \eta_{var} \frac{\nabla f(w)}{\|\nabla f(w)\|}$
- $\eta_{var} = \eta\|\nabla f(w)\|$
- $new\ w = old\ w\ - \eta\|\nabla f(w)\| \frac{\nabla f(w)}{\|\nabla f(w)\|}$
- $new\ w = old\ w\ - \eta\nabla f(w)$



η too small η too large   variable η – just right

Gradient Descent Algorithm

- Initialize the weights.
- While stopping criteria isn't met
  - Compute the gradient $\nabla f(w)$
  - Update the weights: $new\ w = old\ w\ - \eta \nabla f(w)$
- Return the final weights once reached the stopping criteria

Gradient Decent For Logistic Regression

- Unfortunately, there is no closed form solution to maximizing $l(\vec{w})$ with respect to $\vec{w}$.
- Therefore, the common approach is to use gradient *ascent* for *maximizing* $l(\vec{w})$ or equivalently use gradient *decent* for *minimizing* $-l(\vec{w})$.

$$Repeat\ \{$$

$$new\ w_j = old\ w_j - \eta \frac{-\partial log l(\vec{w})}{\partial w_j}$$

$$\}$$

- The j^th component of the vector gradient has the form

$$\frac{-\partial log l(\vec{w})}{\partial w_j} = \sum_{i=1}^{N} (h(\vec{x}^i) - y^i) x_j^i$$

References

- Learning from Data, by Abu-Mustafa, Ismail, Lin
- An Introduction to Statistical Learning with Applications in R, by James, Witten, Hastie, Tibshirani