Lecture 4

# Logistic Regression and Classification

**Physical Address for Prof. Ilias Tagkopoulos**
Computer Science:
Office: 3063 Kemper Hall
Phone: (530) 752-4821
Fax: (530) 752-4767

Genome and Biomedical Sciences Facility:
Office: 5313 GBSF
Phone: (530) 752-7707
Fax: (530) 754-9658

Instructor: Ilias Tagkopoulos
iliast@ucdavis.edu

## From last lecture: Remember the definitions

- You were given a dataset with $m$ samples $D = \{(x^{(i)}, y^{(i)}); i = 1 \dots m\}$.
  - Note that the superscript $x^{(i)}$ is the index of the sample.
  - Assume that each sample has $n$ attributes (features)

$$D = \left\{ \boxed{\begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_n^{(2)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix}}, \boxed{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(m)} \end{bmatrix}} \right\} \text{ or}$$

$$D = \left\{ \boxed{\begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \cdots \\ (x^{(m)})^T \end{bmatrix}}, \boxed{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(m)} \end{bmatrix}} \right\} \text{ with } x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ \cdots \\ x_n^{(i)} \end{bmatrix} \text{ being the } i^{th} \text{ sample}$$

$$D = \{\boxed{X}, \boxed{Y}\}$$

# Previous lecture: Formulating a Regression problem

- By using the dataset D, you can **build a function that relates the input (x) to the output (y)**.
- You can then use this function to **predict** what the output (y) will be, based on a new, possibly never seen before, input.

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}): \boldsymbol{X} \rightarrow \boldsymbol{Y}$$

With **weight vector** $\boldsymbol{w} = \begin{bmatrix} w_0 \\ \cdots \\ w_n \end{bmatrix}$ and **input vector** $\boldsymbol{x}^{(\boldsymbol{i})} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ \cdots \\ x_n^{(i)} \end{bmatrix}$

- The task is to find the optimal parameter values for this function, i.e. the $\boldsymbol{w}$, so that the function $f(x;w)$ ***best describes the relationship between X and Y***

- **Method 1: Ordinary Least Squares**

$$\frac{\partial \boldsymbol{RSS}}{\partial \boldsymbol{w}} = \mathbf{0} \Rightarrow$$

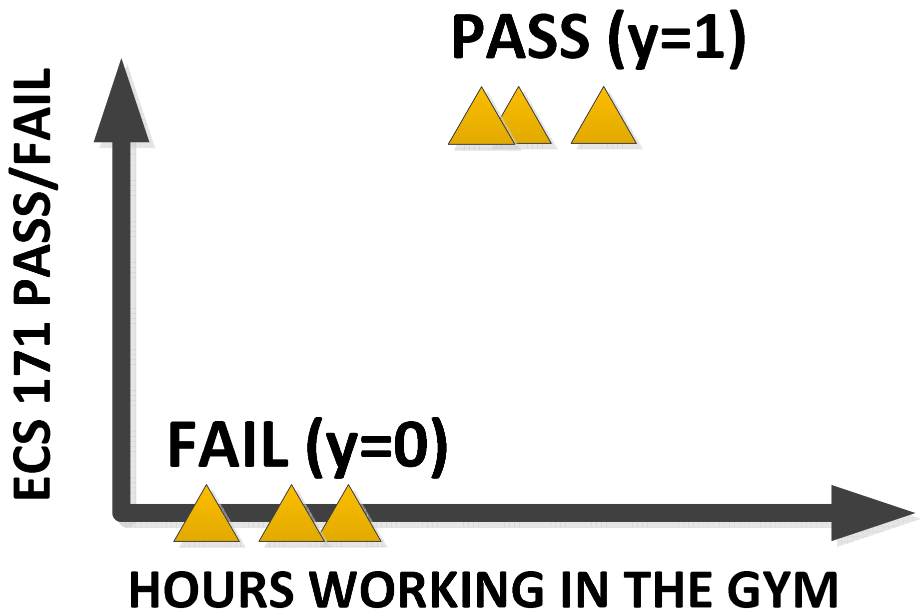
$$w = (X^T X)^{-1} X^T Y$$

- **Method 2: Gradient Descent**

$$\underbrace{\boxed{w_j}}_{\text{Next } w_j} := \underbrace{\boxed{w_j}}_{\text{Previous } w_j} + \underbrace{\boxed{a}}_{\text{constant}} \underbrace{\boxed{(y^{(i)} - \sum_{k=0}^{n} w_k x_k^{(i)}) x_j^{(i)}}}_{\text{Update proportional to error}}$$

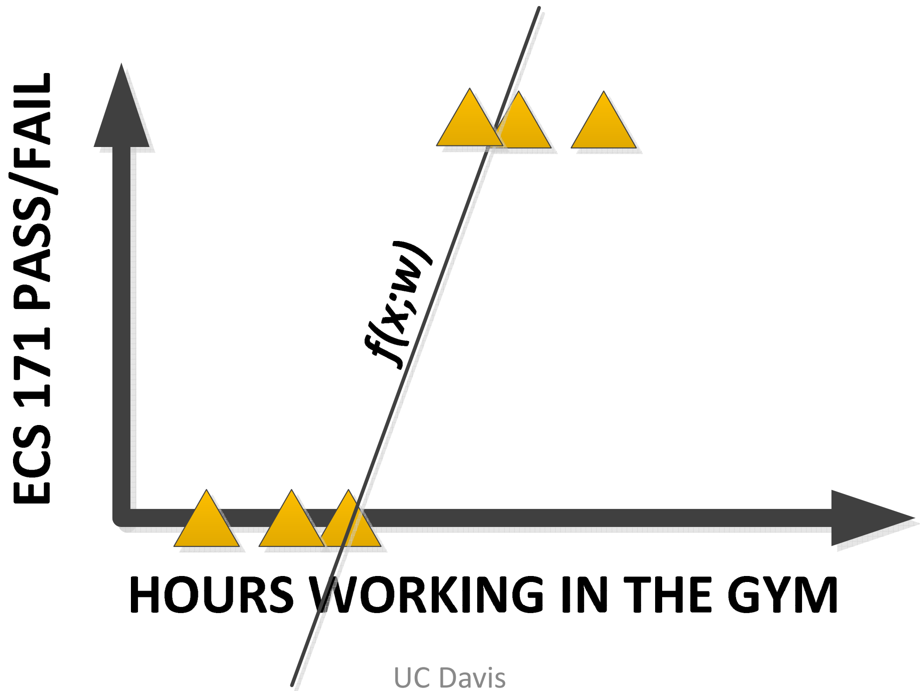
## Logistic Regression: a classification method

- Suppose that all your aim is not to find the value of a continuous variable *per se*, but to categorize the samples into buckets or classes.
- Sure, you can still perform linear regression and then threshold it, but the solution will be sensitive to outliers and to the selected threshold.



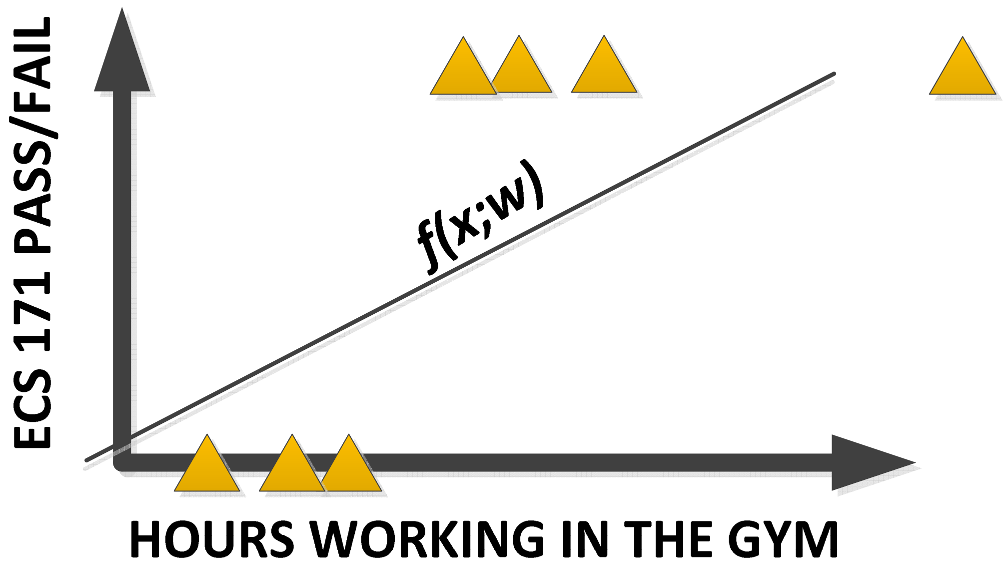
## Logistic Regression: a classification method

- Suppose that all your aim is not to find the value of a continuous variable *per se*, but to categorize the samples into buckets or classes.
- Sure, you can still perform linear regression and then threshold it, but the solution will be sensitive to outliers and to the selected threshold.



## Logistic Regression: a classification method

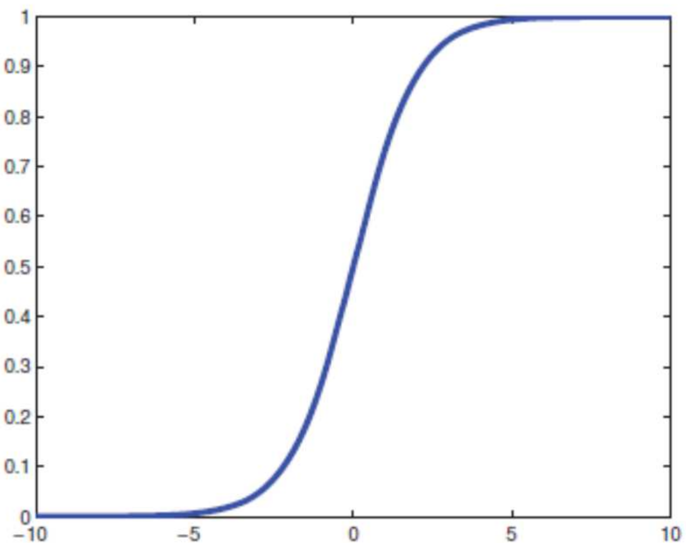
- Suppose that all your aim is not to find the value of a continuous variable *per se*, but to categorize the samples into buckets or classes.
- Sure, you can still perform linear regression and then threshold it, but the solution will be sensitive to outliers and to the selected threshold.



## Logistic Regression: a classification method

- A better way to perform classification when within the regression framework is logistic regression
  - Same formulation as with linear regression, but instead of a polynomial, use the sigmoid/logit/logistic function:

$$sigm(z) = \frac{1}{1 + e^{-z}}$$



# Logistic Regression: a classification method

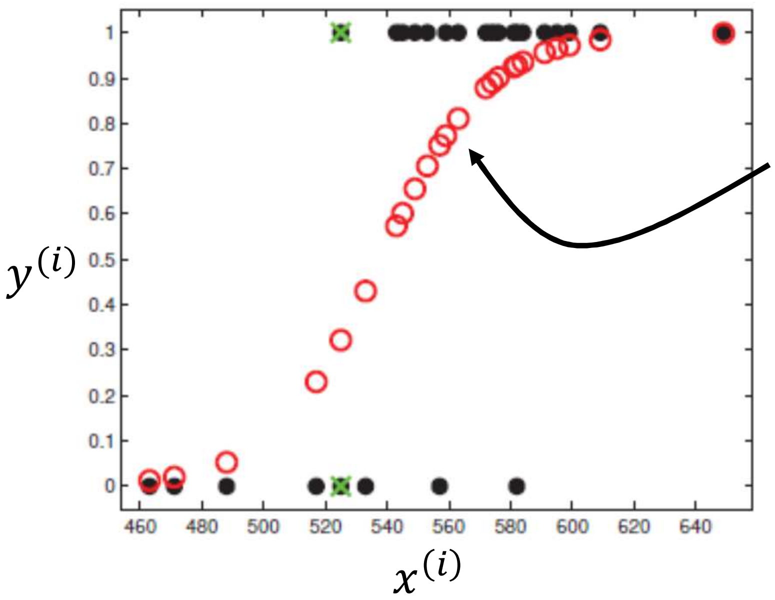
- As such, we can create the function

$$\boldsymbol{g(x; w) = sigm(w^T x) = \frac{1}{1 + e^{-w^T x}}}$$

And categorize into two classes (positive or negative) for any given sample $i$, by defining the label $y^{(i)}$ to be:

$$y^{(i)} = \begin{cases} 0 & if \quad g(w^T x^{(i)}) < threshold \\ 1 & if \quad g(w^T x^{(i)}) \geq threshold \end{cases}$$

With logistic regression, we can express it as probabilities too (instead of hard boundaries)




**Decision boundary**

Are these classes perfectly separable?

## Logistic Regression: MLE

- Great, but how do we find the optimal parameter/weight **w** set?
- There is no closed form solution (such as the OLS for linear regression) but we can first formulate it as a MLE problem and then use gradient descent to find the parameters.
- To do that, lets express the probability of each class:

$$P\left(y^{(i)} = 1 \,\middle|\, x^{(i)}; \boldsymbol{w}\right) = g(x^{(i)}; w)$$
$$P\left(y^{(i)} = 0 \,\middle|\, x^{(i)}; \boldsymbol{w}\right) = 1 - g(x^{(i)}; w)$$

Which can also be written as:

$$p\left(y^{(i)} \middle| x^{(i)}; \boldsymbol{w}\right) = g(x^{(i)}; w)^{y^{(i)}} (1 - g(x^{(i)}; w))^{1-y^{(i)}}$$

## Logistic Regression: MLE

- Assuming the samples are i.i.d. then we can write the log Likelihood as :

$$l(\mathrm{w}) \triangleq logp(D|w) = \sum_{i=1}^{M} \log p(y^{(i)}|x^{(i)};\mathrm{w}) =$$

$$= \sum_{i=1}^{M} \log\left(g(x^{(i)};w)^{y^{(i)}}(1-g(x^{(i)};w))^{1-y^{(i)}}\right) =$$

$$=\sum_{i=1}^{M} y^{(i)}\log g\left(x^{(i)};w\right) + (1-y^{(i)})\log(1-g(x^{(i)};w))$$

- To maximize the log likelihood, we first find the derivative of the log likelihood with respect to w:

$$\frac{\partial l(\mathrm{w})}{\partial w}$$

Which yields

$$\frac{\partial l(w)}{\partial w_j} = \left(y^{(i)} - g\left(x^{(i)};w\right)\right) x_j^{(i)}$$

# Logistic Regression: Gradient Descent

- With that, we can now apply gradient descent (ascent), which as we know updates the w based on the following rule:

$$w_j := w_j + a\frac{\partial l(w)}{\partial w_j}$$

Or

$$w_j := w_j + a\left(y^{(i)} - g(x^{(i)}; w)\right)x_j^{(i)}$$

Have you seen this before?

- Btw, gradient descent is not the only method to find the parameter w. E.g. Newton's method

$$w_j := w_j - \frac{\frac{\partial l(w)}{\partial w_j}}{\frac{\partial^2 l(w)}{(\partial w_j)^2}}$$

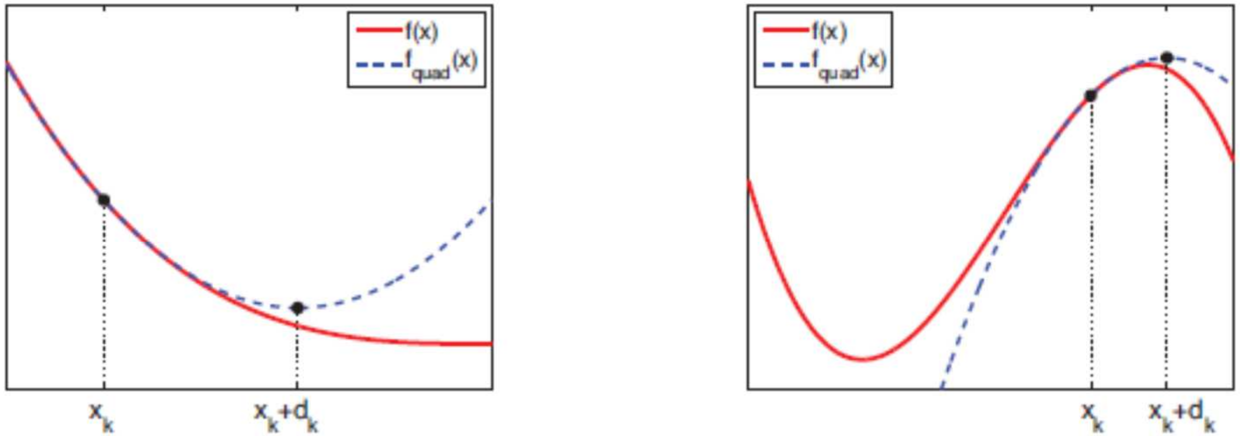
# Logistic Regression: Newton-Raphson

**Algorithm 8.1:** Newton's method for minimizing a strictly convex function

```
Initialize θ_0;
for k = 1, 2, ... until convergence do
    Evaluate g_k = ∇f(θ_k);
    Evaluate H_k = ∇²f(θ_k);
    Solve H_k d_k = −g_k for d_k;
    Use line search to find stepsize η_k along d_k;
    θ_{k+1} = θ_k + η_k d_k;
```

$$w_j := w_j - \frac{\frac{\partial l(w)}{\partial w_j}}{\frac{\partial^2 l(w)}{(\partial w_j)^2}}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \mathbf{H}_k^{-1} \mathbf{g}_k$$




(a) (b)

## Perceptron

- Actually if we use the same update rule but with hard boundaries, forcing the output to be {0,1}, we have the perceptron learning algorithm

$$w_j := w_j + a\left(y^{(i)} - g(x^{(i)}; w)\right)x_j^{(i)}$$

with

$$g(z) = \begin{cases} 0 & if \quad z < threshold \\ 1 & if \quad z \geq threshold \end{cases}$$

Threshold can be any scalar (e.g. 0).

- Generally Stochastic Gradient Descent on logistic regression is faster and has a better performance.

**End of Lecture 4**