Lecture 3

Regression

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

Formulating a Linear Regression problem: **Data**

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

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

Formulating a Linear Regression problem: **Data**

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

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

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

Formulating a Linear Regression problem: **Data**

- 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)

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

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

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

Formulating a Linear Regression problem: **Data**

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

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

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

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

Formulating a Linear Regression problem: **Objective**

- 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.

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

- 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***

Formulating a Linear Regression problem: **Objective**

- But two questions to answer:
  - What structure will this function have ? (assumption)
  - What does "best describes *the relationship between X and Y*" mean?
- In linear regression, we make the assumption that the output is a linear function of the input. So in a perfect word and for any given sample (i), we have:

$$y^{(i)} = f\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right) = \boldsymbol{w}^T \boldsymbol{x}^{(i)} = \sum_{j=0}^{n} w_j x_j^{(i)}$$

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

Formulating a Linear Regression problem: **Objective**

- In an imperfect world, however, we usually cannot find one set of **w**'s so the output $y^{(i)}$ is exactly expressed as a linear function of the input $\boldsymbol{x}^{(i)}$ for each sample *i*. We can define this difference as the residual error $\epsilon^{(i)}$:

$$y^{(i)} = f\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right) + \epsilon^{(i)} = \boldsymbol{w}^T \boldsymbol{x}^{(i)} + \epsilon^{(i)} = \sum_{j=0}^{n} w_j x_j^{(i)} + \epsilon^{(i)}$$

Or equivalently:

$$\epsilon^{(i)} = (\text{real value of } y^{(i)}) - (\text{predicted } y^{(i)} \text{ value}) \rightarrow$$
$$\epsilon^{(i)} = y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)}$$

To have an absolute measure of the error for all samples in the dataset, we can have the error $\epsilon^{(i)}$ squared and take the sum over all ***m*** samples (which we called **Residual Sum of Squares, RSS**).

$$\boldsymbol{RSS} = \sum_{i=1}^{m} (\epsilon^{(i)})^2 = \sum_{i=1}^{m} (y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)})^2$$

Formulating a Linear Regression problem: **Objective**

- If we minimize the residual squared error, by selecting the appropriate weights **w**, ***we describe the relationship between x and y in the best possible way***, given our assumptions.
- In the previous lecture, we ended up with the same result from a Bayesian perspective, proving that ***minimizing the RSS is equivalent to maximizing the log-likelihood of the data, given the model*****.**
- As such, our task is:

$$\boldsymbol{w} \triangleq \underset{\boldsymbol{w}}{\operatorname{argmin}}\, RSS = \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)} \right)^2 \right)$$

- We will go through two ways to do that, analytically in a method called **Ordinary Least Squares (OLS)** and numerically with **gradient descent.**

Solving the problem: **Ordinary Least Squares (Method 1)**

- We first have to express the RSS in a matrix form. Note that

$$(\boldsymbol{Y - Xw}) = \underset{m \times 1}{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(m)} \end{bmatrix}} - \underset{m \times n}{\begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_n^{(2)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix}} \underset{n \times 1}{\begin{bmatrix} w_0 \\ \cdots \\ w_n \end{bmatrix}} =$$

$$= \begin{bmatrix} y^{(1)} - \sum_{j=0}^{n} w_j x_j^{(1)} \\ y^{(2)} - \sum_{j=0}^{n} w_j x_j^{(2)} \\ \cdots \\ y^{(m)} - \sum_{j=0}^{n} w_j x_j^{(m)} \end{bmatrix} = \underset{m \times 1}{\begin{bmatrix} y^{(1)} - \boldsymbol{w}^T \boldsymbol{x}^{(1)} \\ y^{(2)} - \boldsymbol{w}^T \boldsymbol{x}^{(2)} \\ \cdots \\ y^{(m)} - \boldsymbol{w}^T \boldsymbol{x}^{(3)} \end{bmatrix}}$$

Solving the problem: **Ordinary Least Squares (Method 1)**

- We will use the fact that for any vector A, we have

$$\boldsymbol{A^T A = \sum_i a_i^2}$$

- Similarly here we have:

$$\boldsymbol{(Y - Xw)^T (Y - Xw)} = \left(\begin{bmatrix} y^{(1)} - \boldsymbol{w^T x^{(1)}} \\ y^{(2)} - \boldsymbol{w^T x^{(2)}} \\ \cdots \\ y^{(m)} - \boldsymbol{w^T x^{(m)}} \end{bmatrix} \right)^{\boldsymbol{T}} \begin{bmatrix} y^{(1)} - \boldsymbol{w^T x^{(1)}} \\ y^{(2)} - \boldsymbol{w^T x^{(2)}} \\ \cdots \\ y^{(m)} - \boldsymbol{w^T x^{(m)}} \end{bmatrix}$$

Or

$$\boldsymbol{(Y - Xw)^T (Y - Xw)} = \boldsymbol{\sum_{i=1}^{m} (y^{(i)} - w^T x^{(i)})^2}$$

Solving the problem: **Ordinary Least Squares (Method 1)**

- How do we find the w's that minimize RSS?
- We differentiate the RSS with respect to **w**, and set it to **zero**:

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

$$\nabla_{\boldsymbol{w}}[(\boldsymbol{Y} - \boldsymbol{Xw})^{\boldsymbol{T}}(\boldsymbol{Y} - \boldsymbol{Xw})] = 0 \Rightarrow$$

$$2X^T(Y - Xw) = 0 \Rightarrow$$

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

OLS solution of Linear Regression

Solving the problem: **Gradient Descent (Method 2)**

- Another way to find the weights that minimize RSS is to use an iterative method that is called **gradient descent (GD)**.
- **Gradient descent**, in general, is GREAT:
  - You don't need to know matrix algebra!
  - It is easy and intuitive to understand.
  - It perform stochastic optimization (heuristic) that can be much faster than an exact algorithm (e.g. branch-and-bound methods) and will always give you an answer (it can be a bad one though).
- Of course nothing is free:
  - GD can take many cycles to converge, or even never converge (batch GD always does).
  - It provides no guarantee regarding the optimality or the bounds of the solution.

Solving the problem: **Gradient Descent (Method 2)**

- In **GD**, the parameters are updated based on the following rule:

$$w_j := w_j - a\frac{\partial \boldsymbol{RSS}}{\partial w_j}$$

- What it basically means is: move w towards the direction that will **minimize the RSS**.
  - If the derivative of RSS with respect to w_j is positive, then higher values of w_j will increase RSS, so decrease w!
  - If the derivative of RSS with respect to w_j is negative, then higher values of w will decrease RSS, so increase w_j!
  - The parameter a is the step of increase/decrease and is called learning rate.

Solving the problem: **Gradient Descent (Method 2)**

- In our case, application of the **GD** update rule in the case of **one sample** yields:

$$w_j := w_j - a\frac{\partial \boldsymbol{RSS}}{\partial w_j} \Rightarrow$$

$$w_j := w_j - a\frac{\partial (y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)})^2}{\partial w_j} \Rightarrow$$

$$w_j := w_j - a\frac{\partial (y^{(i)} - \sum_{k=0}^{n} w_k x_k^{(i)})^2}{\partial w_j} \Rightarrow$$

$$w_j := w_j + 2a(y^{(i)} - \sum_{k=0}^{n} w_k x_k^{(i)}) x_j^{(i)}$$

Next w_j Previous w_j constant Update proportional to error

Solving the problem: **Gradient Descent (Method 2)**

- This is called also the **Least Mean Squares (LMS) update rule** (or **Widrow-Hoff** learning rule). Lets omit the "2" for simplicity as it can be part of a and write:

$$w_j := w_j + a\,(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)})x_j^{(i)}$$

- What if we have ***m samples*** **?** Two ways to deal with it:

Batch gradient descent

Repeat until convergence:
{ for j=1 to n

$$w_j := w_j + a \sum_{i=1}^{m}(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)})x_j^{(i)}$$

}

Stochastic gradient descent

Repeat until convergence:
{ for i= 1 to m
{ for j=1 to n

$$w_j := w_j + a(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)})x_j^{(i)}$$

}
}

Solving the problem: **Gradient Descent (Method 2)**

$$f(\boldsymbol{\theta}) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2,$$

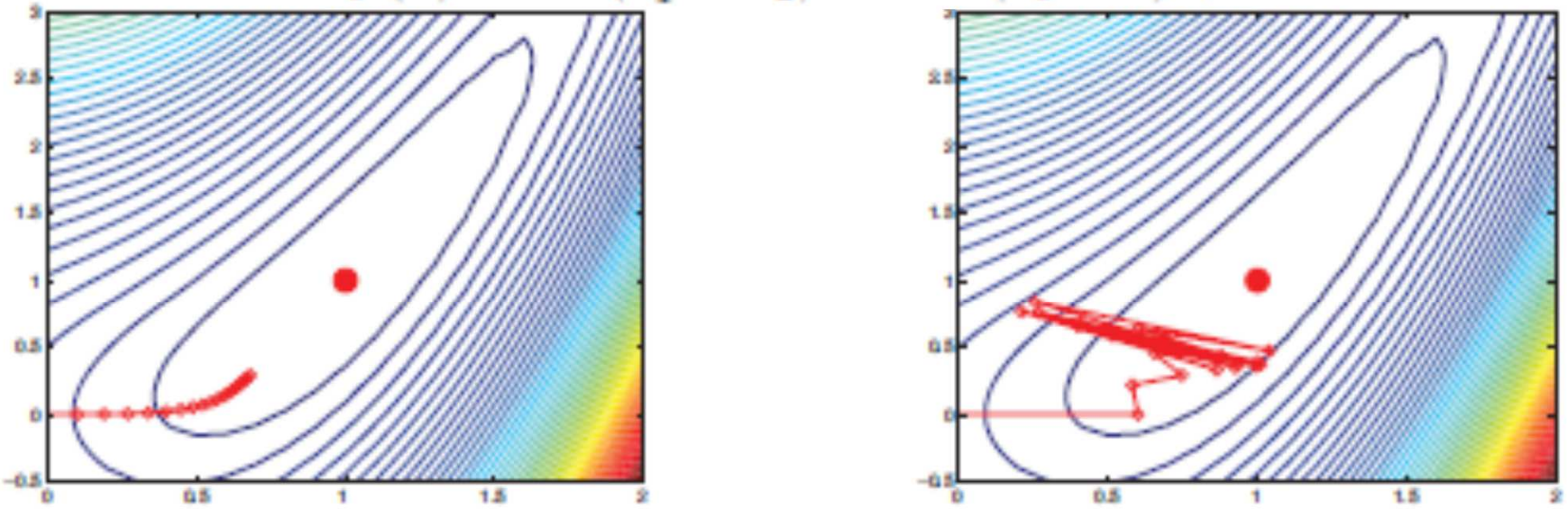
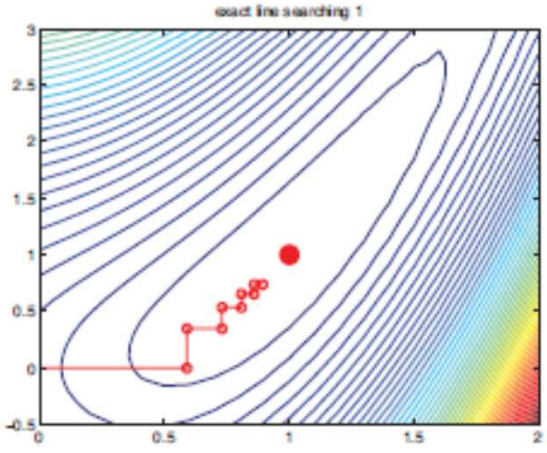


Figure 8.2 Gradient descent on a simple function, starting from $(0, 0)$, for 20 steps, using a fixed learning rate (step size) η. The global minimum is at $(1, 1)$. (a) $\eta = 0.1$. (b) $\eta = 0.6$. Figure generated by `steepestDescentDemo`.



End of Lecture 3