Numerical Optimization for ML&DL (NOFML&DL)

Artificial Intelligence and Data Science

شرح بالعربي

1

Agenda

GD Applied to Multivariate LR.

Features Scaling.

x Area	y Price
8450	208500
9600	181500
11250	223500
9550	140000
14260	250000
14115	143000
10084	307000

x_0 Bias (intersect) multiplier	x_1 Income	x_2 House Age	x_3 Number of Rooms	x_4 Number of Bedrooms	y Price (e+06)
1	79545	5	7	4	1.059
1	79248	6	6	3	1.505
1	61287	5	8	5	1.058
1	63345	7	5	3	1.260
1	59982	5	7	4	6.309

GD Applied to Multivariate LR

- **Single Variable LR:** $h_\theta(x) = \theta_0 + \theta_1 x$
- **Multi Variable LR**
- $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$
- $x_0 = 1$
- $h_\theta(x) = \Theta^T X$

GD Applied to Multivariate LR

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

GD Applied to Multivariate LR

Gradient Descent

Previously (n=1):

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial\theta_0}J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \ldots, n$)

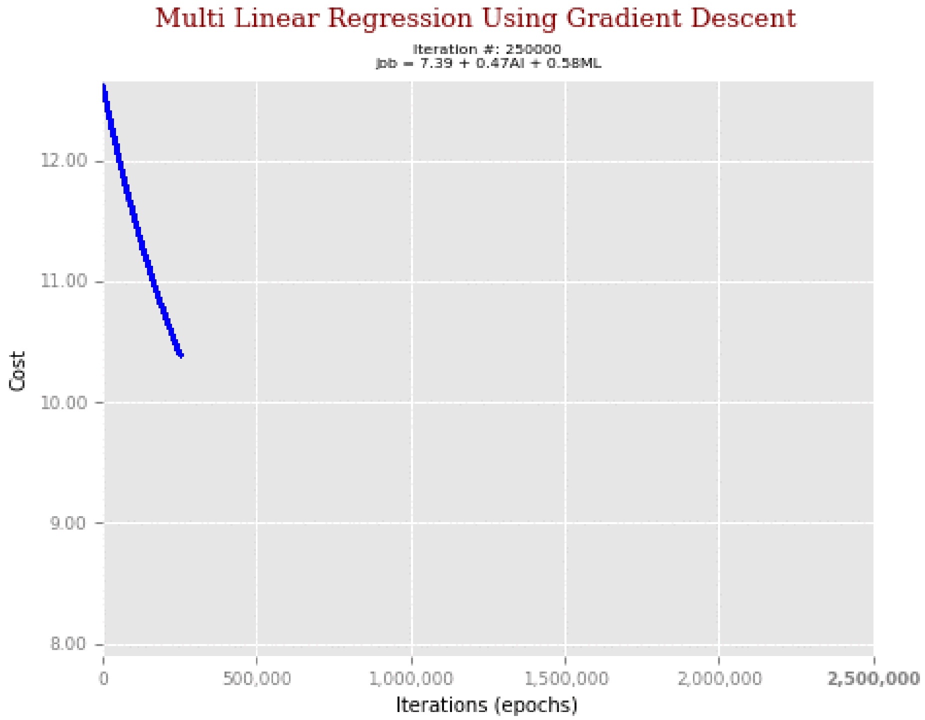
$\}$

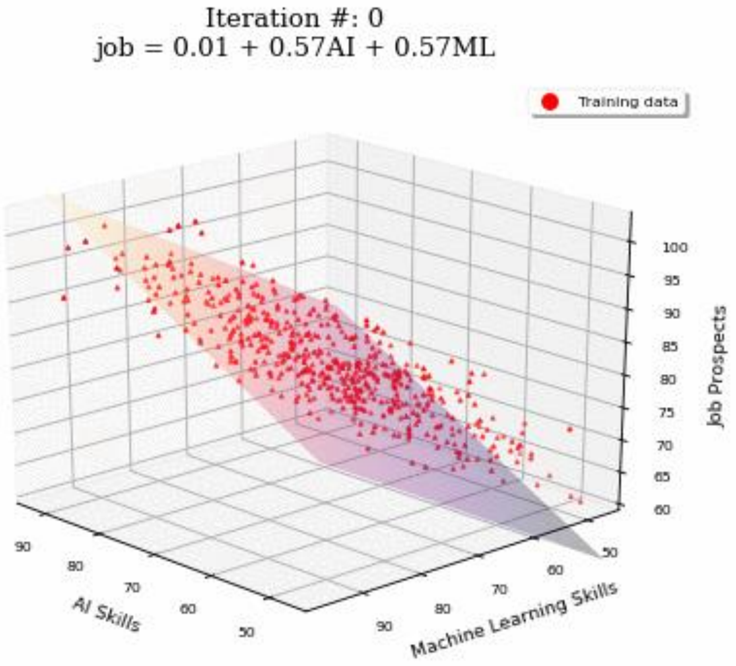
$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$$

$\ldots$

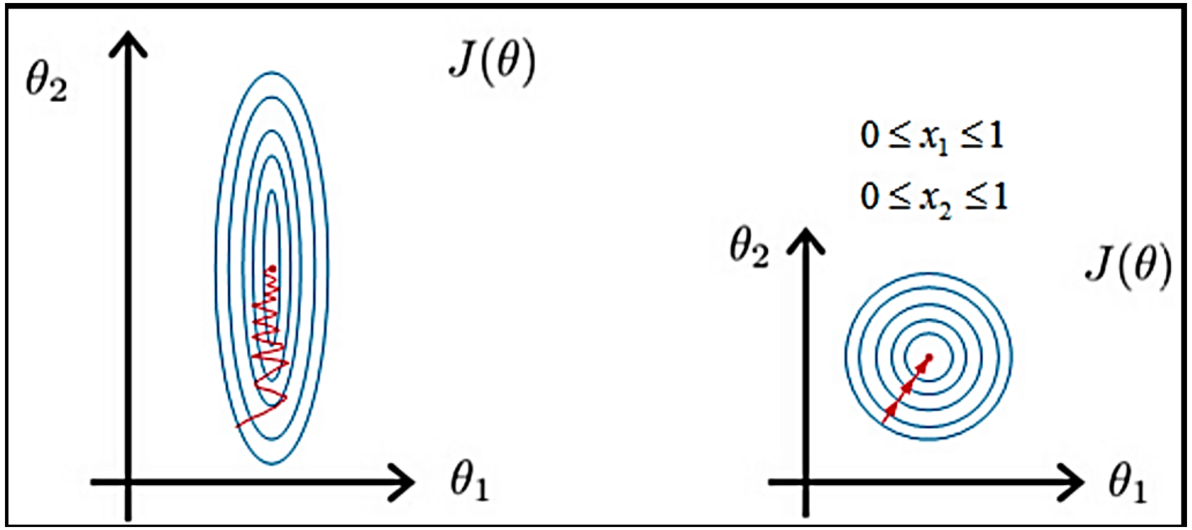




GD Applied to Multivariate LR

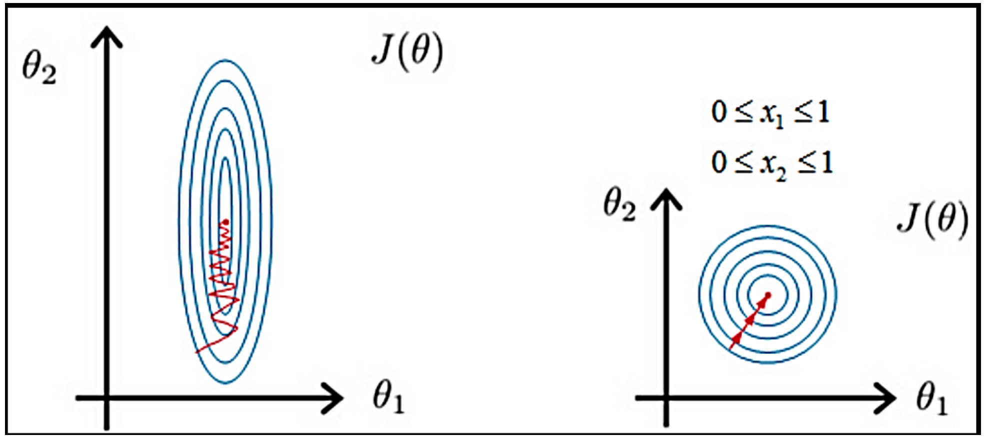
Features Scaling:

- **Features Scaling:**
  - Make sure features are on similar scale.
  - For gradient-based algorithms, features scaling improves the convergence speed.



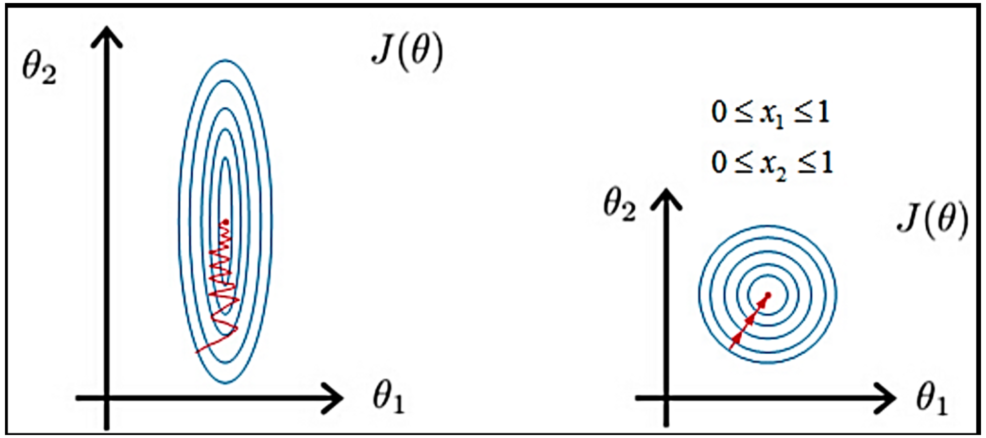
Features Scaling:

- **Problem Statement:**
  - If we have two features $\boldsymbol{x_1}$ with large scale and $\boldsymbol{x_2}$ with low scale the equivalent $\boldsymbol{\theta_1}$ will be small and have a small search range and $\boldsymbol{\theta_2}$ will be large and has large search range.
  - In the opposite side, as long as the gradient vector components depends on the feature value, the gradient component for the large-scale feature will be larger than the small.



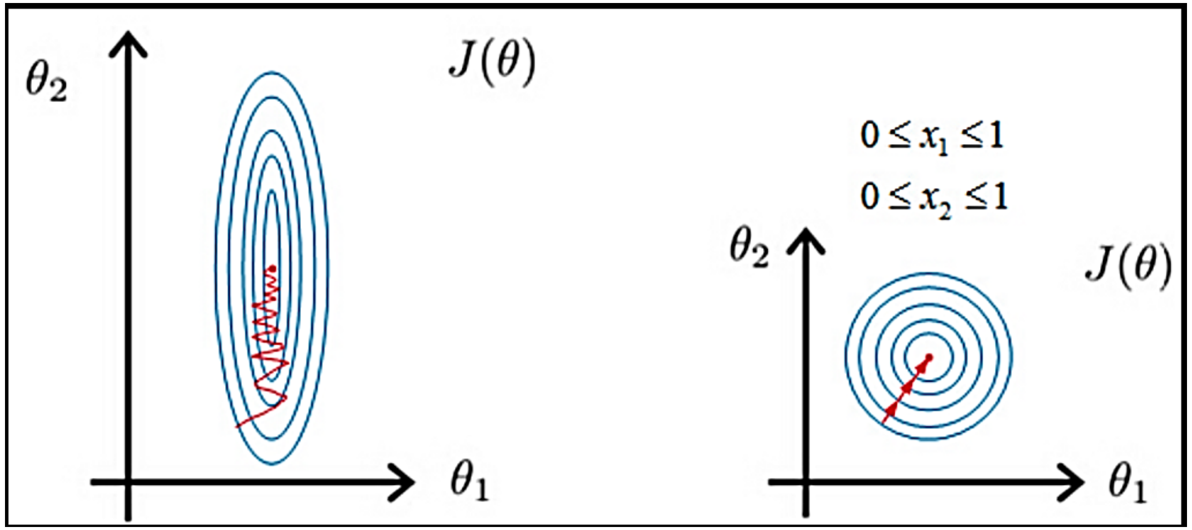
Features Scaling:

- **Features Statement:**
  - This implies a small range of $\boldsymbol{\theta_1}$ with large update in its direction and large range of $\boldsymbol{\theta_2}$ with small update in its direction.
  - This makes the gradient descent oscillates during training and consumes large number of iterations.



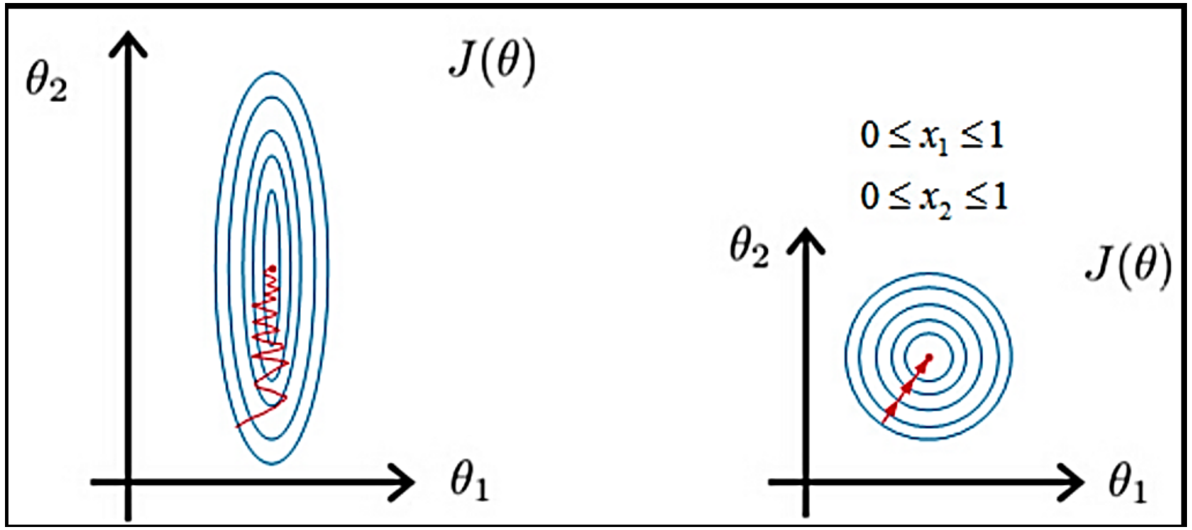
Features Scaling:

- Make sure features are on similar scale.
- For gradient-based algorithms, features scaling improves the convergence speed.
- Distance-based algorithms like **KNN, K-means,** and **SVM** are most affected by the range of features.
- **Tree-based** algorithms, on the other hand, are fairly **insensitive** to the scale of the features.



Features Scaling:

- Use feature scaling when the algorithm calculates distances (K-Nearest Neighbor and Support Vector Machines) or is trained with Gradient Descent (Regression).



Features Scaling:

- **Min-Max Normalization:**
  (Sometimes just called normalization)
  - It scales each variable/feature in the [0,1] range.
  - This method preserves the shape of the original distribution and is sensitive to outliers.

```
from sklearn.preprocessing import MinMaxScaler
```

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Features Scaling:

- **Mean Normalization:** (Sometimes just called standardization)
  - It produces a distribution centered at 0 with a standard deviation of 1.
  - This method "makes" a feature normally distributed. With outliers, the data will be scaled to a small interval.

```
from sklearn.preprocessing import StandardScaler
```

$$x' = \frac{x - \bar{x}}{\sigma}$$

Features Scaling:

- **Robust Scaling**
  - All distributions have most of their densities around 0 and a shape that is more or less the same.
  - The Interquartile range makes this method robust to outliers (hence the name).

```python
from sklearn.preprocessing import RobustScaler
```

$$x' = \frac{x - Q_2(x)}{Q_3(x) - Q_1(x)}$$

, where Q are quartiles.

Batch/Vanilla GD

- **The main advantages:**
  - We can use fixed learning rate during training without worrying about learning rate decay.
  - It has straight trajectory towards the minimum and it is guaranteed to converge in theory to the global minimum if the loss function is convex and to a local minimum if the loss function is not convex.
  - It has unbiased estimate of gradients. The more the examples, the lower the standard error.
- **The main disadvantages:**
  - Even though we can use vectorized implementation, it may still be slow to go over all examples especially when we have large datasets.
  - Each step of learning happens after going over all examples where some examples may be redundant and don’t contribute much to the update.

Resources

- https://www.coursera.org/learn/machine-learning
- https://machinelearningmastery.com/analytical-vs-numerical-solutions-in-machine-learning/
- https://www.youtube.com/watch?v=e6kf6DDQVYA&ab_channel=TreeSoftMatterTheory
- https://en.wikipedia.org/wiki/Mathematical_optimization
- https://builtin.com/data-science/gradient-descent
- https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
- https://math.stackexchange.com/questions/2202545/why-using-squared-distances-in-the-cost-function-linear-regression
- https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-ii-d20a239cde11
- https://www.mathsisfun.com/gradient.html
- https://en.wikipedia.org/wiki/Derivative
- https://www.mathsisfun.com/calculus/derivatives-introduction.html
- https://math.libretexts.org/Bookshelves/Calculus/Map%3A_Calculus__Early_Transcendentals_(Stewart)/14%3A_Partial_Derivatives/14.01%3A_Functions_of_Several_Variables
- https://slideplayer.com/slide/4753135/
- https://en.wikipedia.org/wiki/Gradient
- https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/partial-derivative-and-gradient-articles/a/the-gradient
- https://la.mathworks.com/help/matlab/ref/meshc.html
- http://www.adeveloperdiary.com/data-science/how-to-visualize-gradient-descent-using-contour-plot-in-python/
- https://rpubs.com/mgswiss15/M6C_7Multivariate
- https://stats.stackexchange.com/questions/354046/coordinate-descent-with-constraints
- https://www.mathworks.com/help/optim/ug/local-vs-global-optima.html#:~:text=A%20local%20minimum%20of%20a,at%20all%20other%20feasible%20points.
- https://en.wikipedia.org/wiki/Maxima_and_minima
- https://wngaw.github.io/linear-regression/
- http://www.cheerml.com/saddle-points
- https://towardsdatascience.com/understand-convexity-in-optimization-db87653bf920

- https://towardsdatascience.com/understand-convexity-in-optimization-db87653bf920

- https://www.sciencedirect.com/topics/engineering/convex-function
- https://www.math24.net/convex-functions#example2
- https://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx
- https://en.wikipedia.org/wiki/Newton's_method

- https://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx

Resources

- https://realpython.com/linear-regression-in-python/
- https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2
- https://towardsdatascience.com/why-norms-matters-machine-learning-3f08120af429
- https://towardsdatascience.com/why-norms-matters-machine-learning-3f08120af429
- https://machinelearningmastery.com/vector-norms-machine-learning/
- https://medium.com/linear-algebra/part-18-norms-30a8b3739bb
- https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
- Andrew Ng, Machine Learning, Stanford University, Coursera
- https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
- https://medium.com/data-science-365/linear-regression-with-gradient-descent-895bb7d18d52
- https://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html
- https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220
- https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220
- https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/
- https://builtin.com/data-science/gradient-descent
- https://www.mltut.com/stochastic-gradient-descent-a-super-easy-complete-guide/
- https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931
- https://kaigangi72.medium.com/stochastic-gradient-descent-demystified-part-1-8e4b897079b7
- https://medium.datadriveninvestor.com/gradient-descent-algorithm-b4c5afb4eb98
- https://medium.com/mindorks/an-introduction-to-gradient-descent-7b0c6d9e49f6
- https://medium.com/@venkatavinay222/at-the-end-machine-learning-is-all-about-optimization-ft-gradient-descent-e1588b7d95d2
- “Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow” by Aurélien Géron
- https://laptrinhx.com/feature-scaling-why-and-how-3308094292/
- https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3