

DS 4400

Machine Learning and Data Mining I
Spring 2024

David Liu

Khoury College of Computer Science
Northeastern University

Friday January 26, 2024

Today's Outline

- Practical considerations for Multiple Linear Regression
- Gradient descent
 - General optimization algorithm
 - Instantiation for linear regression
 - Issues with gradient descent
 - Comparison with closed-form solution
- Non-linear regression
 - Polynomial regression
 - Cubic, spline regression

Federal Trade Commission Launches Inquiry Into A.I. Deals by Tech Giants




The agency plans to scrutinize Microsoft, Amazon and Google for their investments in the A.I. start-ups OpenAI and Anthropic.



Satya Nadella, Microsoft's chief executive, right, and Sam Altman, the head of OpenAI, which makes ChatGPT. Microsoft has invested billions of dollars in OpenAI. Justin Sullivan/Getty Images

As part of Thursday's inquiry, the F.T.C. said, it will ask Microsoft, Amazon, Google's parent company, Alphabet, OpenAI and Anthropic for details including whether the deals between the giants and the start-ups involved have rights to board seats or other oversight over each other. Microsoft [obtained](#) a seat on OpenAI's board in November but does not have the right to vote on its decisions.

Microsoft has committed \$13 billion for what is effectively a 49 percent stake in the start-up. The company worked to keep its share below 50 percent because of antitrust concerns, The New York Times previously reported. Amazon said it would invest [up to \\$4 billion in Anthropic](#). Google has committed to investing more than \$2 billion in Anthropic.

	Lina M. Khan Chair Sworn in: June 15, 2021 Biography Speeches, Articles, & Testimony Twitter
	Rebecca Kelly Slaughter Commissioner Sworn in: May 2, 2018 Biography Speeches, Articles, & Testimony Twitter
	Alvaro Bedoya Commissioner Sworn in: May 16, 2022 Biography Speeches, Articles, & Testimony Twitter

Supplemental Materials

Materials for multivariable calculus
(computing partial derivatives and the chain rule)

<https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives>

Row Reduction

<https://people.tamu.edu/~f-narcowich//psfiles/rowred2012.pdf>

A note on proofs

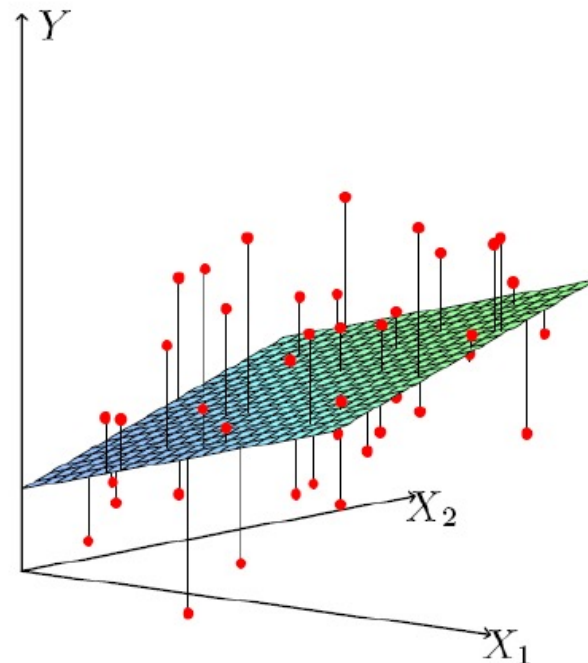
Multiple Linear Regression

- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_{\theta}(x) = \theta^T x$
- $MSE = \frac{1}{N} \sum (\theta^T x_i - y_i)^2$ **Loss / cost**

$$\theta = (X^T X)^{-1} X^T y$$

Closed-form optimum
solution for linear regression

X and y are given data



PRACTICAL CONSIDERATIONS

Practical issues: Feature Standardization

- Rescales features to have zero mean and unit variance

- Let μ_j be the mean of feature j:

$$\mu_j = \frac{1}{N} \sum_{i=1}^N x_{ij}$$

- Replace each value with:

$$x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{s_j} \quad \text{for } j = 1 \dots d$$

(not x_0 !)

- s_j is the standard deviation of feature j

Other feature normalization

- Min-Max rescaling

- $x_{ij} \leftarrow \frac{x_{ij} - \min_j}{\max_j - \min_j}$

- \min_j and \max_j : min and max value of feature j

- Mean normalization

- $x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{\max_j - \min_j}$

Other feature normalization

- Min-Max rescaling

- $x_{ij} \leftarrow \frac{x_{ij} - \min_j}{\max_j - \min_j} \in [0,1]$

- \min_j and \max_j : min and max value of feature j

- Mean normalization

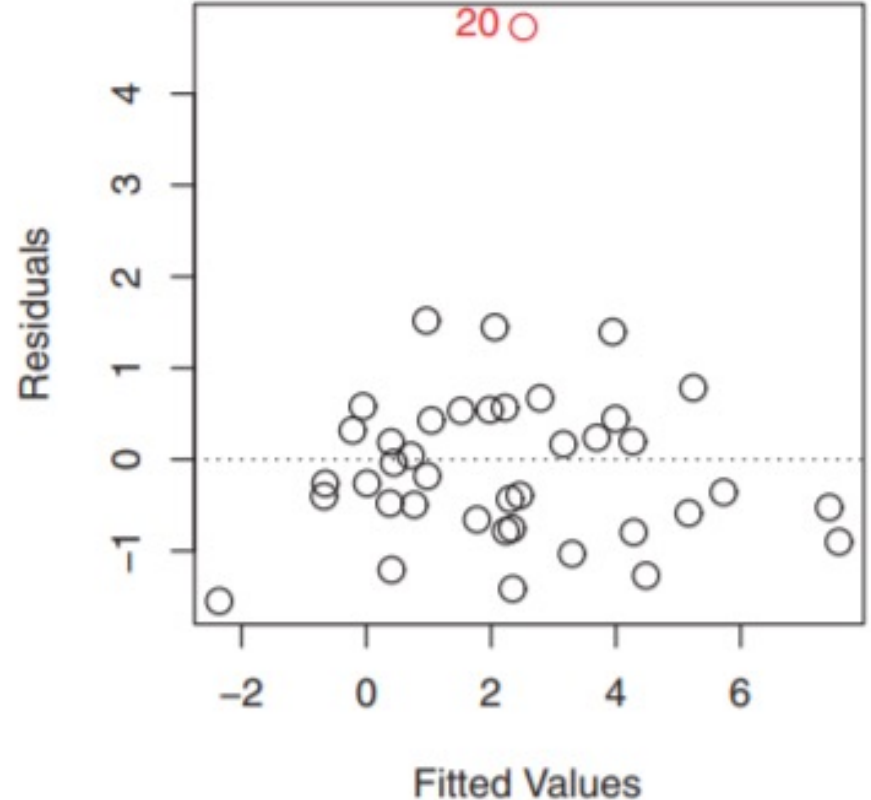
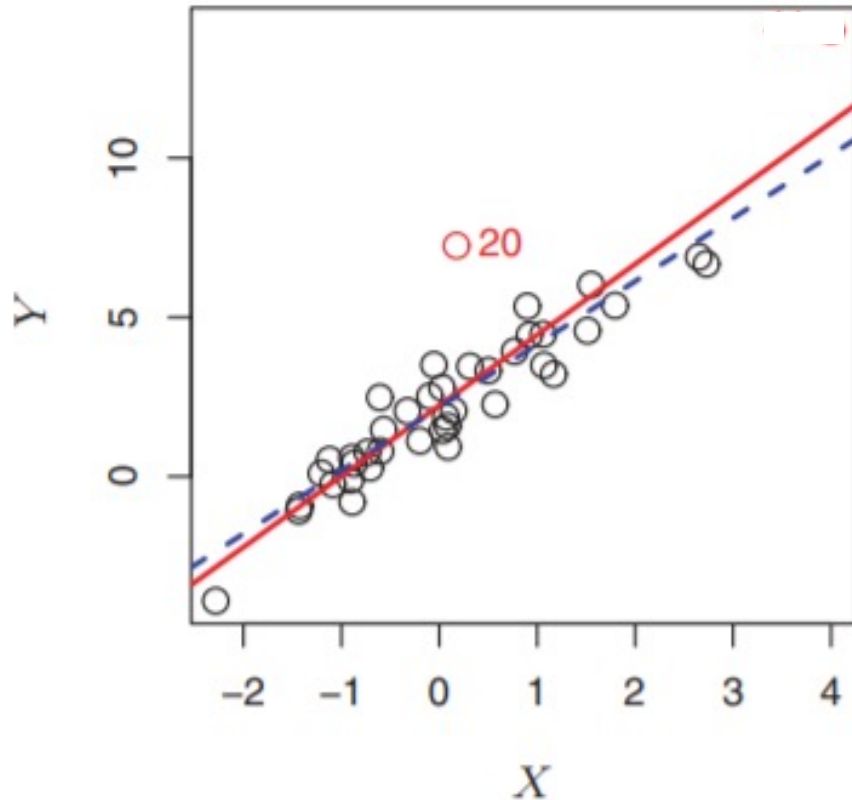
- $x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{\max_j - \min_j}$

- Mean 0

Feature standardization/normalization

- Goal is to have individual features on the same scale
- Is a pre-processing step in most learning algorithms
- Necessary for linear models and Gradient Descent
- Different options:
 - Feature standardization
 - Feature min-max rescaling
 - Mean normalization

Practical issues: Outliers



- Dashed model is without outlier point
- Linear regression is not resilient to outliers!
- Outliers can be eliminated based on residual value
 - Use different loss functions (Huber loss)

Practical issues: Categorical features

- Predict credit card balance
 - Age
 - Income
 - Number of cards
 - Credit limit
 - Credit rating
- Categorical variables
 - Student (Yes/No)
 - State (50 different levels)

How to generate numerical representations of these?

Indicator Variables

- One-hot encoding
- Binary (two-level) variable
 - Add new feature $x_j = 1$ if student and 0 otherwise
- Multi-level variable
 - State: 50 values
 - $x_{MA} = 1$ if State = MA and 0, otherwise
 - $x_{NY} = 1$ if State = NY and 0, otherwise
 - ...
 - How many indicator variables are needed?
- Disadvantages: data becomes too sparse for large number of levels
 - Will discuss feature selection later in class

Recap Linear Regression

- Optimal solution to min MSE
 - Use vectorization for compact representation
- Advantages of linear regression
 - Simplicity and interpretability
 - Closed-form optimal solution (depends uniquely on training data)
- Limitations of linear regression
 - Small capacity in number of parameters ($d+1$)
 - Does not fit well non-linear data
- Practical issues
 - Feature standardization
 - Outliers
 - Categorical features

Training phase of most ML

- Input: labeled data
- Define objective / loss metric
 - MSE for regression
 - Specific loss functions for classification
- Run an optimization procedure to minimize loss (error) on training data
- Output: trained model that best fits the training data

How to optimize loss functions?

- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- $J(\theta) = \frac{1}{N} \sum (\theta^T x_i - y_i)^2$ Loss / cost for regression
- General method to optimize a multi-variate function
 - Practical and efficient
 - Generally applicable to different loss functions
 - Convergence guarantees for certain loss functions (e.g., convex)

What Strategy to Use?



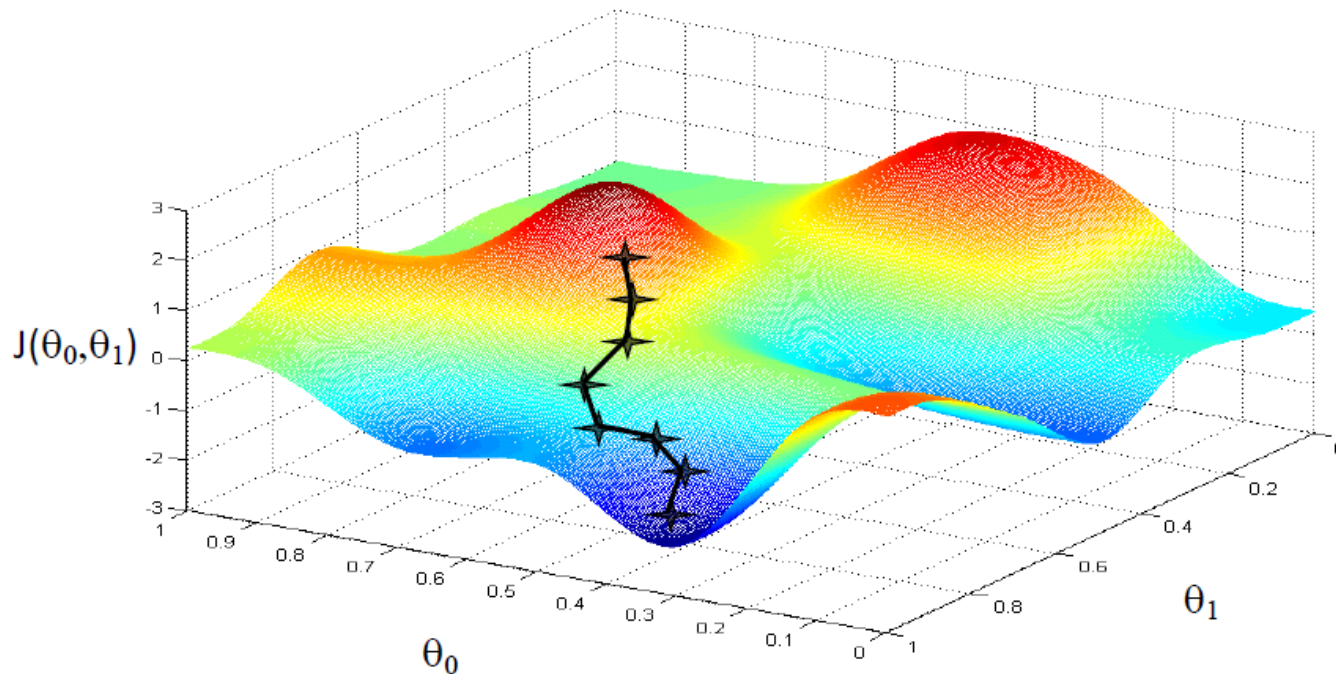
Follow the Slope



Follow the direction of steepest descent!

How to optimize $J(\theta)$?

- Choose initial value for θ
- Until we reach a minimum:
 - Choose a new value for θ to reduce $J(\theta)$



Gradient Descent

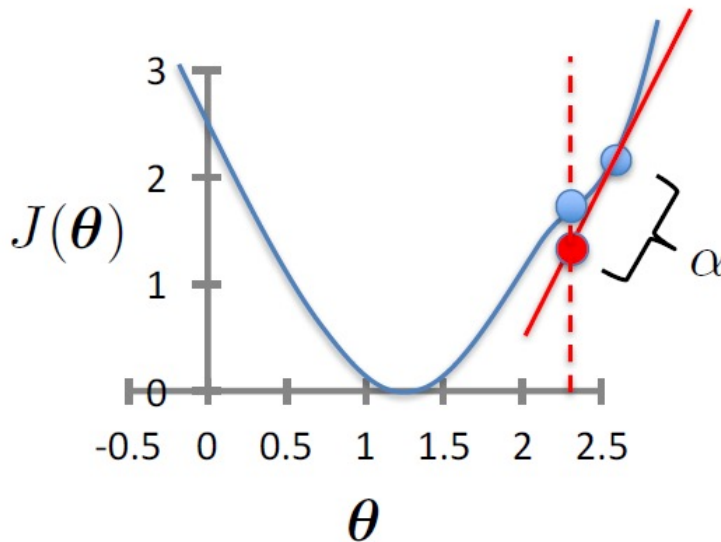
Gradient Descent

- Initialize θ
- Repeat until convergence

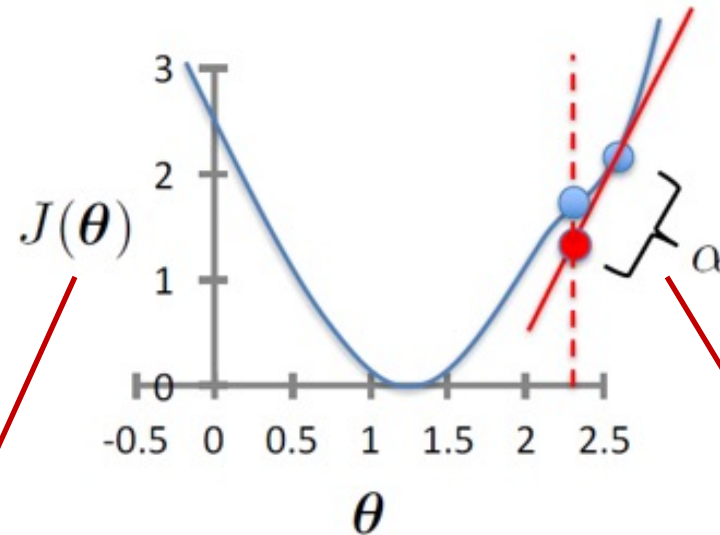
$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

learning rate (small)
e.g., $\alpha = 0.05$



Gradient Descent



- If θ is on the left of minimum, slope is negative
- Increase value of θ

- If θ is on the right of minimum, slope is positive
- Decrease value of θ

In both cases θ gets closer to minimum

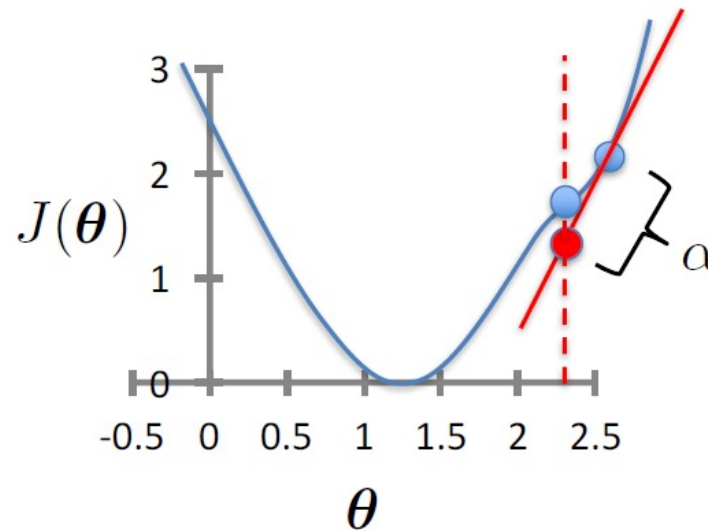
Stopping Condition

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

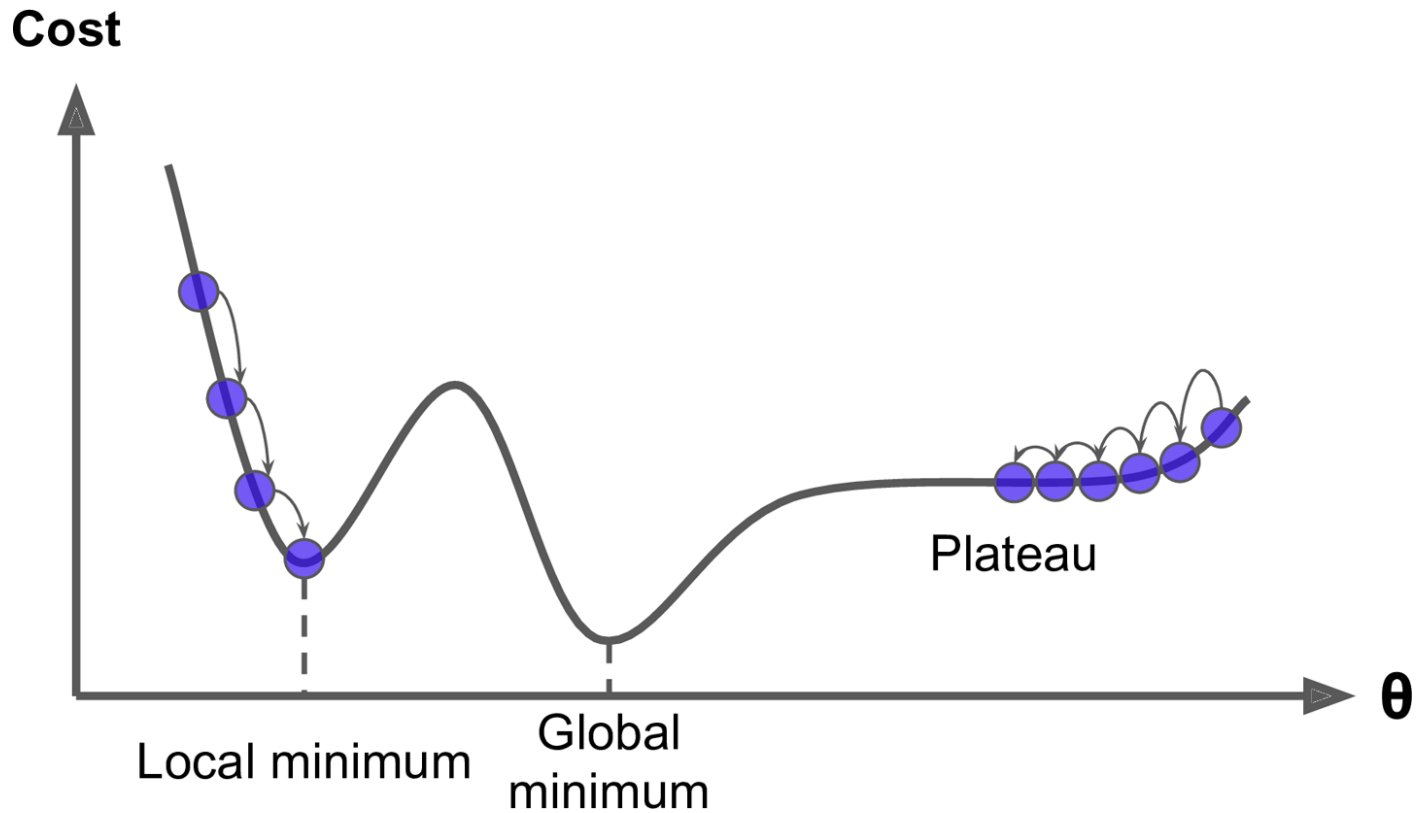
simultaneous update
for $j = 0 \dots d$

learning rate (small)
e.g., $\alpha = 0.05$



- When should the algorithm stop?

GD Convergence Issues



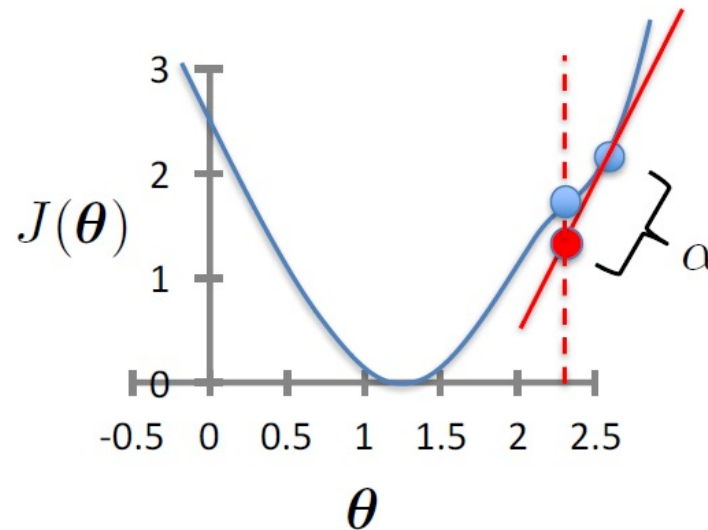
Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

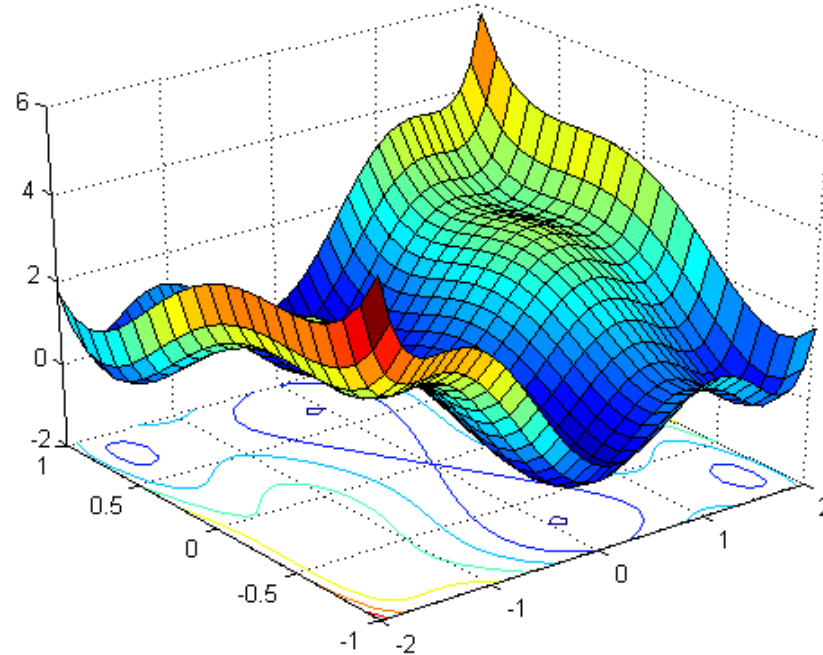
simultaneous update
for $j = 0 \dots d$

learning rate (small)
e.g., $\alpha = 0.05$



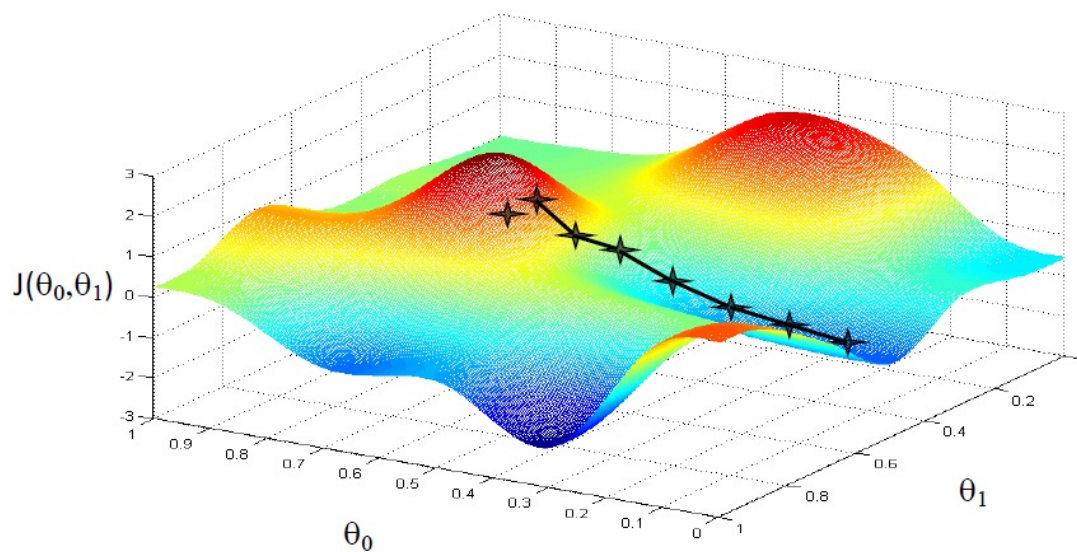
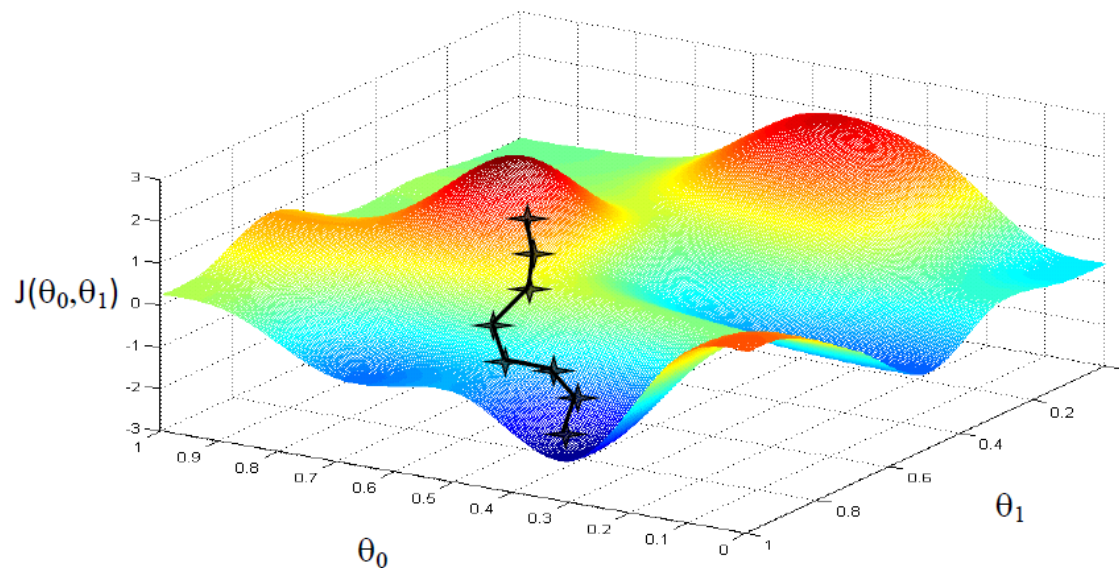
- What happens when θ reaches a local minimum?

Complex loss function



- Convex loss functions only have global minimum, no local minima
- Complex loss functions are more difficult to optimize as they have multiple local optima

Possible Solution



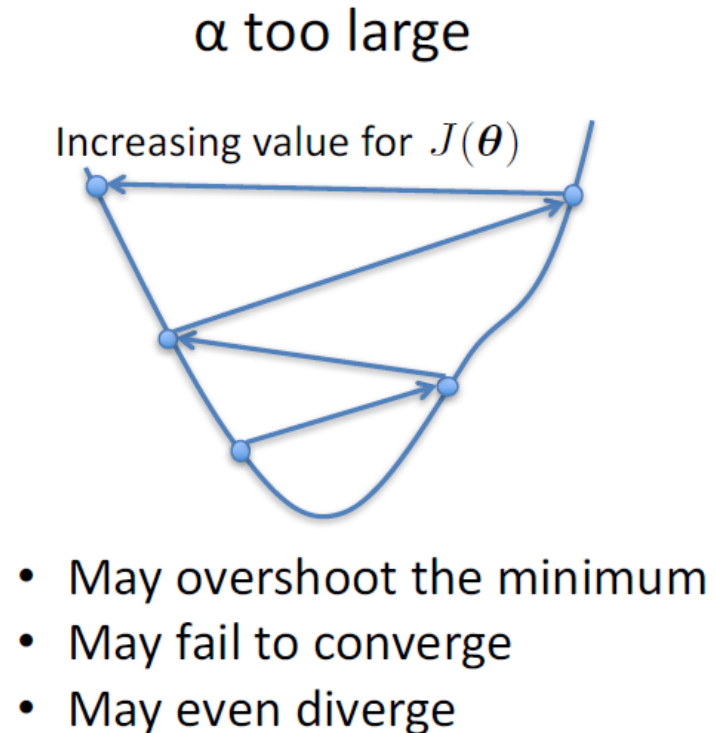
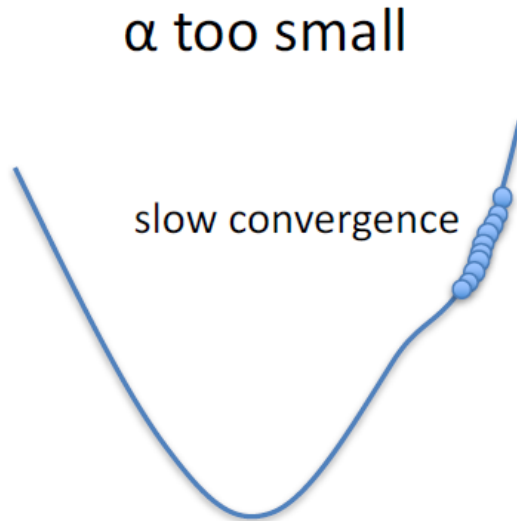
Adjusting Learning Rate

α too small

α too large



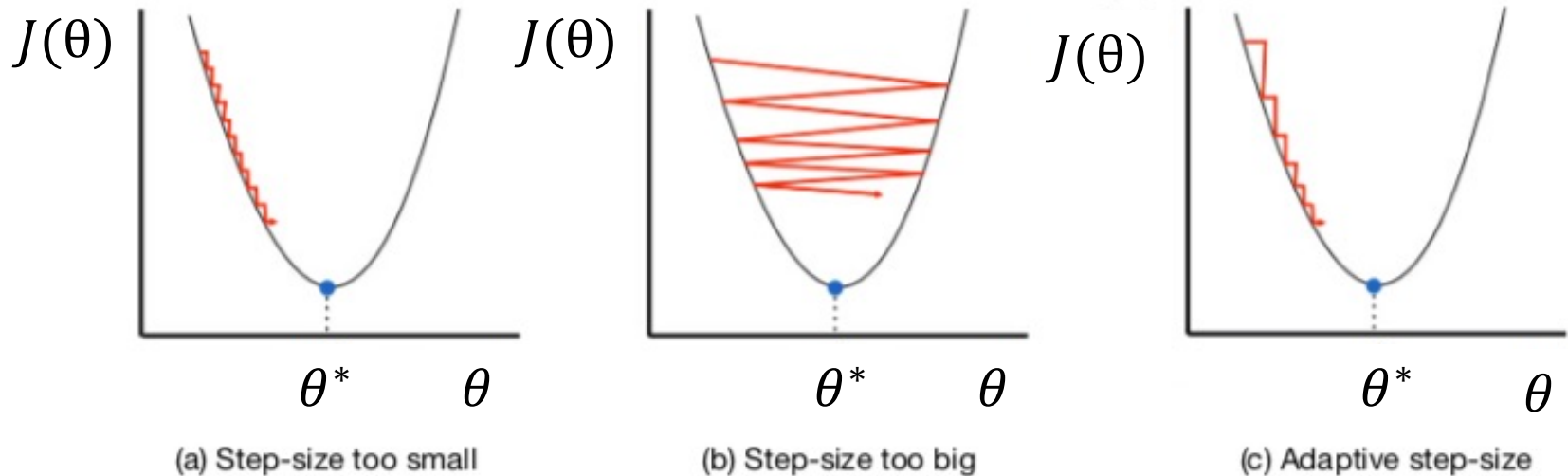
Choosing Learning Rate



To see if gradient descent is working, print out $J(\theta)$ each iteration

- The value should decrease at each iteration
- If it doesn't, adjust α

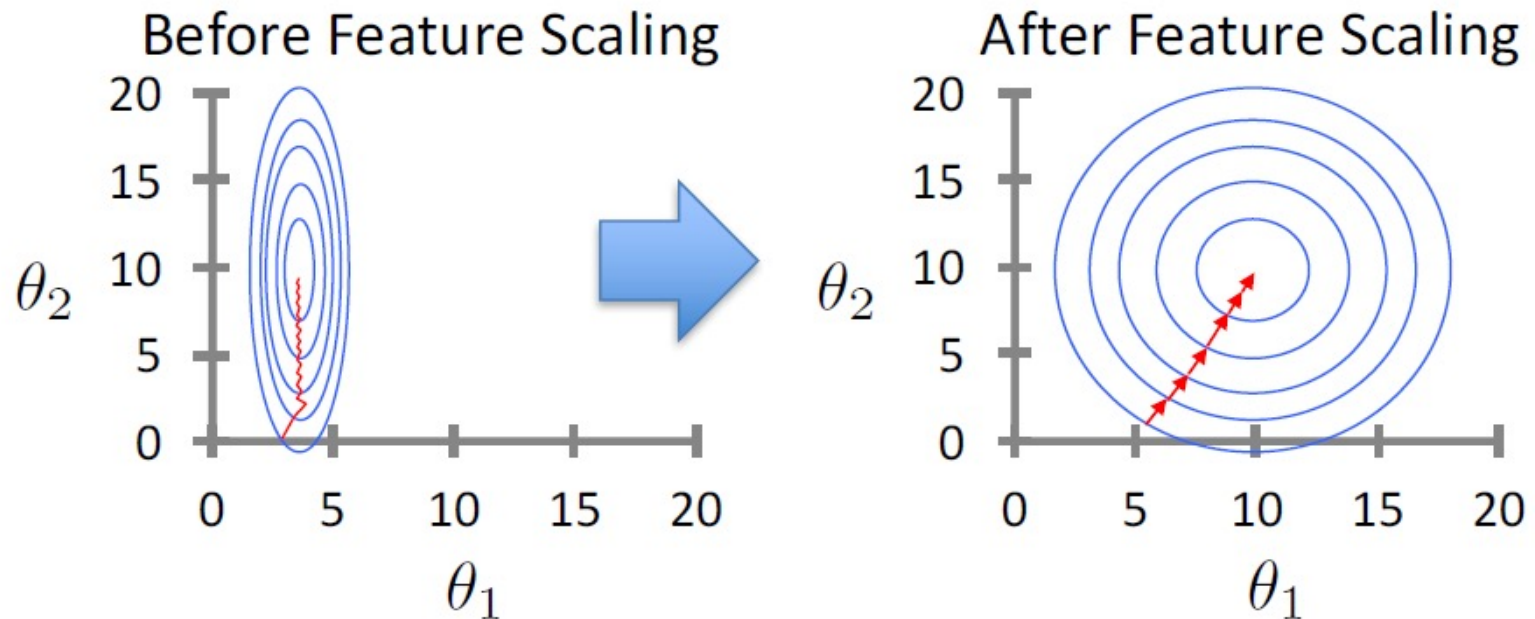
Adaptive step size



- Start with large step size and reduce over time, adaptively
- Line search method
- Measure how objective decreases

Feature Scaling

- **Idea:** Ensure that feature have similar scales



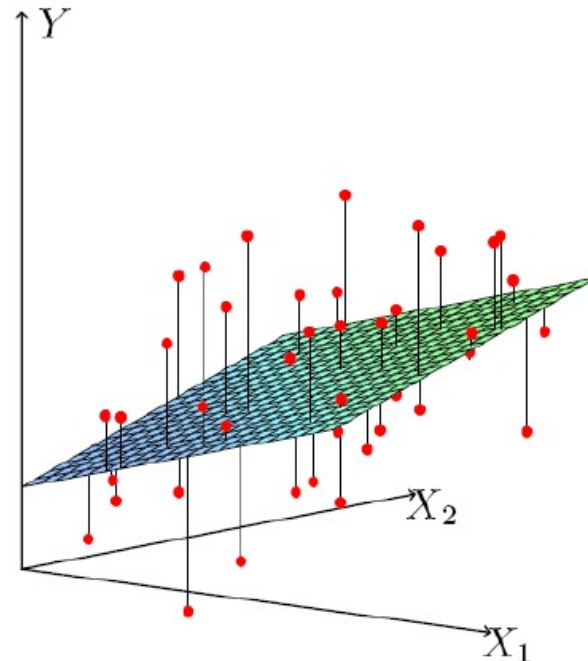
- Makes gradient descent converge *much* faster

Multiple Linear Regression

- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- $MSE = \frac{1}{N} \sum (h_\theta(x_i) - y_i)^2$ **Loss / cost**

$$\theta = (X^T X)^{-1} X^T y$$

**MSE is a strictly convex function
and has unique minimum**



GD for Multiple Linear Regression

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

GD for Linear Regression

- Initialize θ
- Repeat until convergence $\|\theta_{new} - \theta_{old}\| < \epsilon$ or $iterations == MAX_ITER$

$$\theta \leftarrow \theta - \alpha \frac{2}{N} (X\theta - y)^T X$$

- To achieve simultaneous update
 - At the start of each GD iteration, compute $h_{\theta}(x_i)$
 - Use this stored value in the update step loop
- Assume convergence when $\|\theta_{new} - \theta_{old}\|_2 < \epsilon$

$$\text{L}_2 \text{ norm: } \|v\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|v|}^2}$$

GD for Linear Regression

- Initialize θ
- Repeat until convergence $\|\theta_{new} - \theta_{old}\| < \epsilon$ or $iterations == MAX_ITER$

$$\theta \leftarrow \theta - \alpha \frac{2}{N} (X\theta - y)^T X$$

Equivalent

$$\theta_j \leftarrow \theta_j - \alpha \frac{2}{N} \sum_{i=1}^N (h_{\theta}(x_i) - y_i) x_{ij}, j = 0, \dots, d$$

- Assume convergence when $\|\theta_{new} - \theta_{old}\|_2 < \epsilon$

$$\text{L}_2 \text{ norm: } \|v\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|v|}^2}$$

Gradient Descent in Practice

- Asymptotic complexity
 - $O(NTd)$, N is size of training data, d is feature dimension, and T is number of iterations
- Most popular optimization algorithm in use today
- At the basis of training
 - Linear Regression
 - Logistic regression
 - SVM
 - Neural networks and Deep learning
 - Stochastic Gradient Descent variants

Gradient Descent vs Closed Form

Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

Closed form solution for LR

$$\theta = (X^T X)^{-1} X^T y$$

- Gradient Descent

- Closed Form

Gradient Descent vs Closed Form

Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

Closed form

$$\theta = (X^T X)^{-1} X^T y$$

• Gradient Descent

- + Linear increase in d and N
- + Generally applicable
- Need to choose α and stopping conditions
- Might get stuck in local optima

• Closed Form

- + No parameter tuning
- + Gives the global optimum
- Not generally applicable
- Slow computation: $O(d^3)$

Issues with Gradient Descent

- Might get stuck in local optimum and not converge to global optimum
 - Restart from multiple initial points
- Only works with differentiable loss functions
- Small or large gradients
 - Feature scaling helps
- Tune learning rate
 - Can use line search for determining optimal learning rate

Review Gradient Descent

- Gradient descent is an efficient algorithm for optimization and training ML models
 - The most widely used algorithm in ML!
 - Faster than using closed-form solution for linear regression
 - Main issues with Gradient Descent is convergence and getting stuck in local optima (for neural networks)
- Gradient descent is guaranteed to converge to optimum for strictly convex functions if run long enough