UNIVERSITY OF ARKANSAS

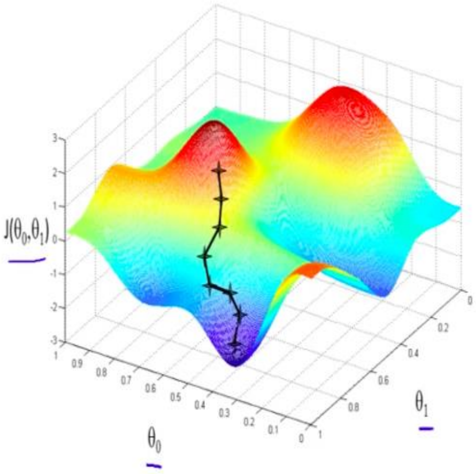
Gradient Descent

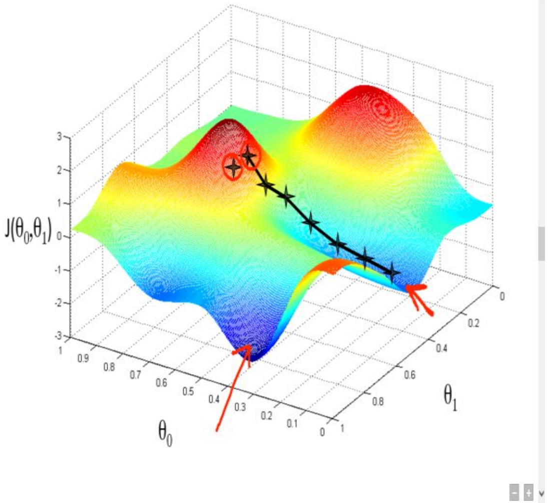
Jiahui Chen
Department of Mathematical Sciences
University of Arkansas

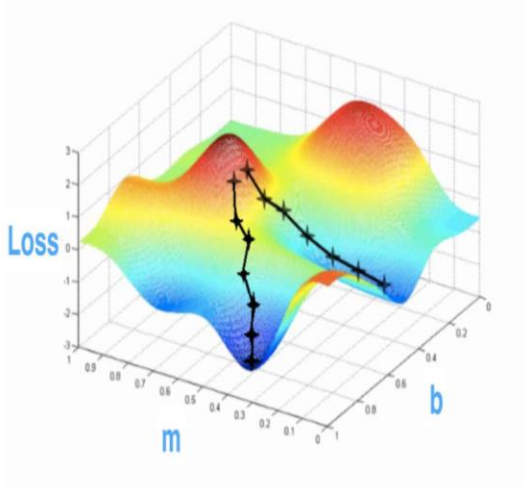
UNIVERSITY OF ARKANSAS

Introduction

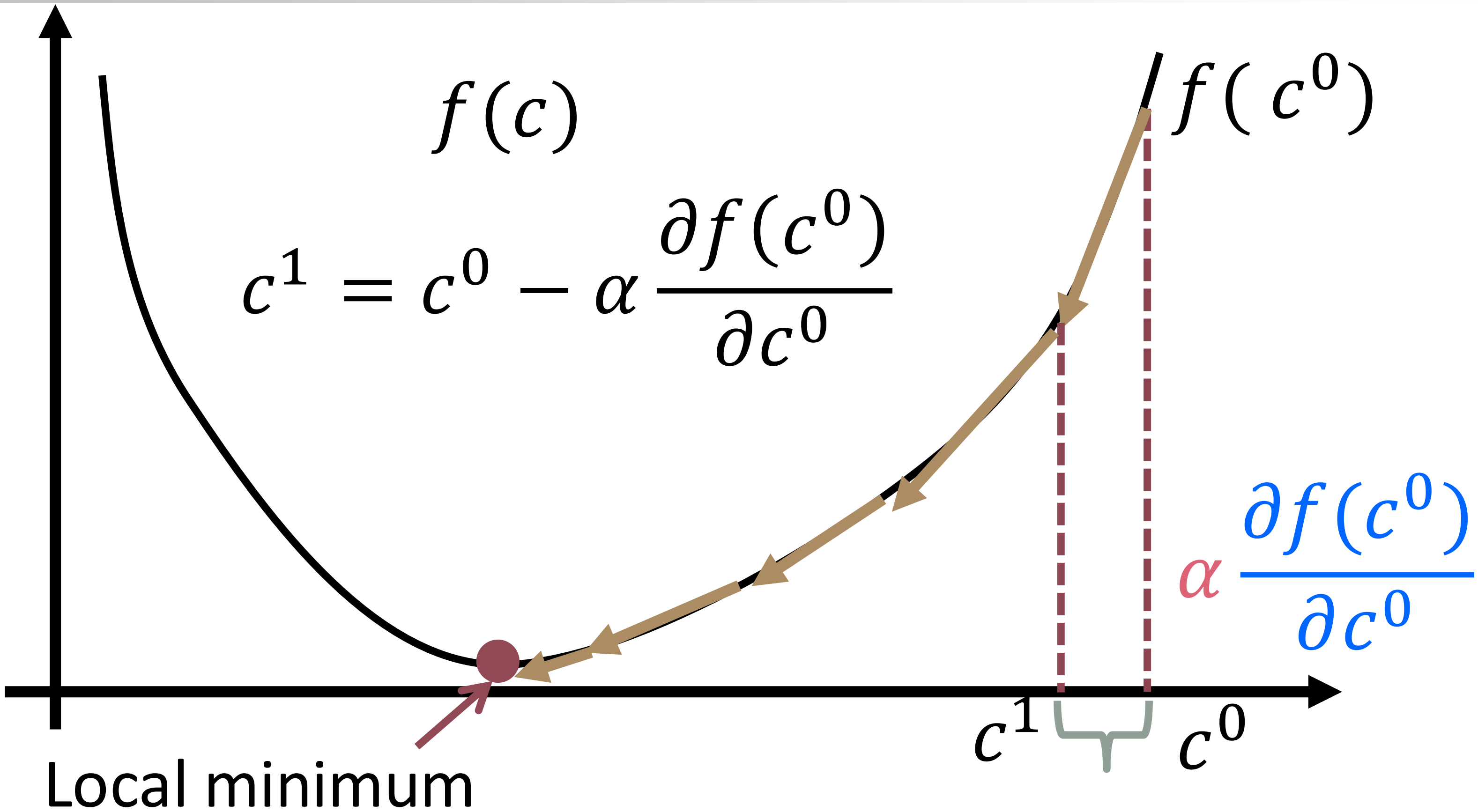
- In general, the loss function has no analytical solutions.
- Gradient = direction of the steepest ascent
- Find a local minimum of a function
- Often a first-order iterative optimization algorithm







General Idea



Algorithm

Find a local minimum of a C^1 continuous $f(c)$

- Start with random value c^0
- Update new value:

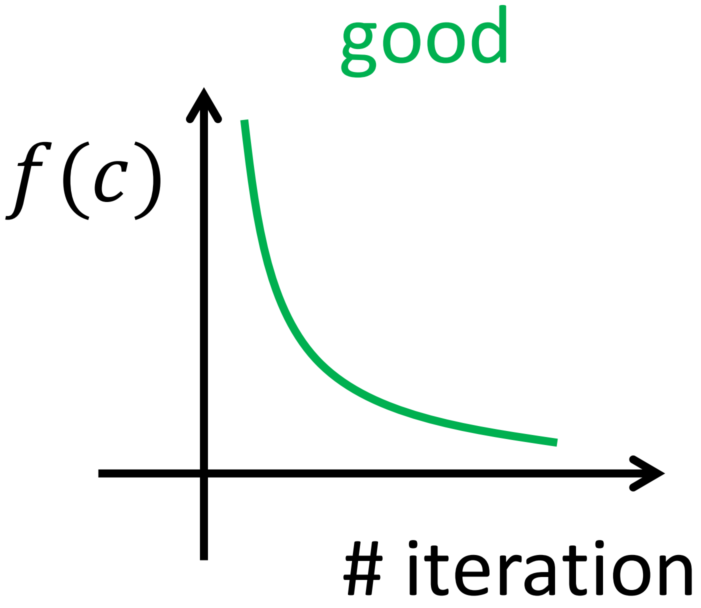
$$c^{i+1} = c^i - \alpha \frac{\partial f(c^i)}{\partial c^i}$$

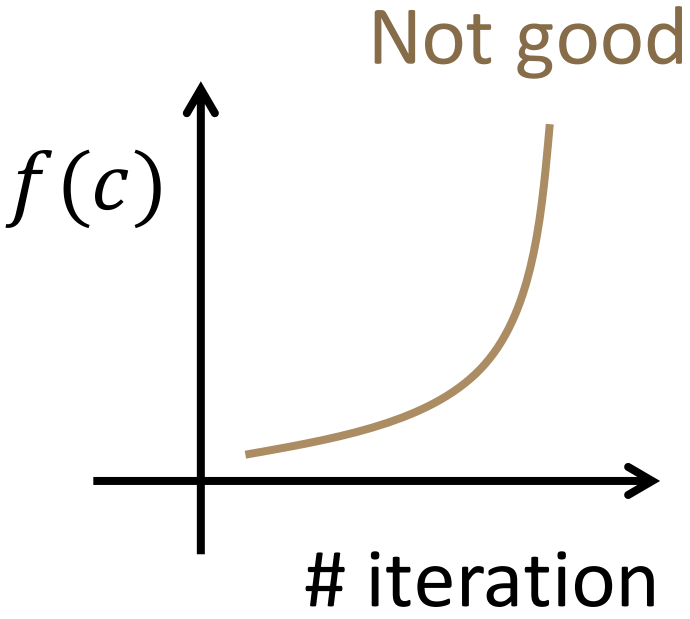
α: **learning rate**, very small

- Repeat until $\left\| \frac{\partial f(c^i)}{\partial c^i} \right\| \leq$ tolerance

Making Sure Gradient Descent Working Correctly

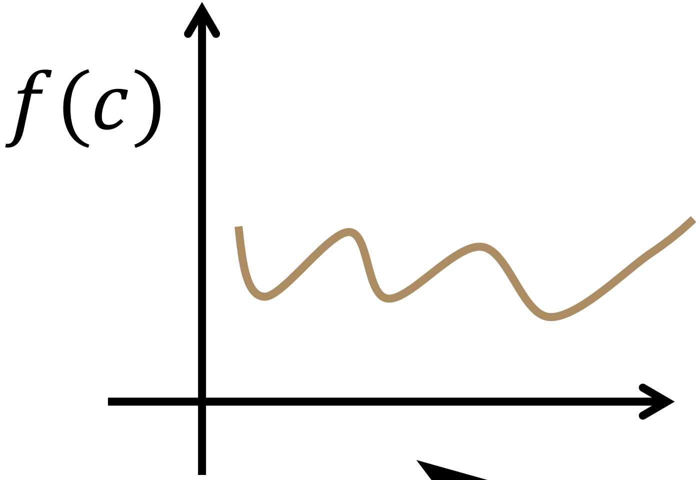
- Function $f(c)$ should decrease after every iteration (monotonically decreases)

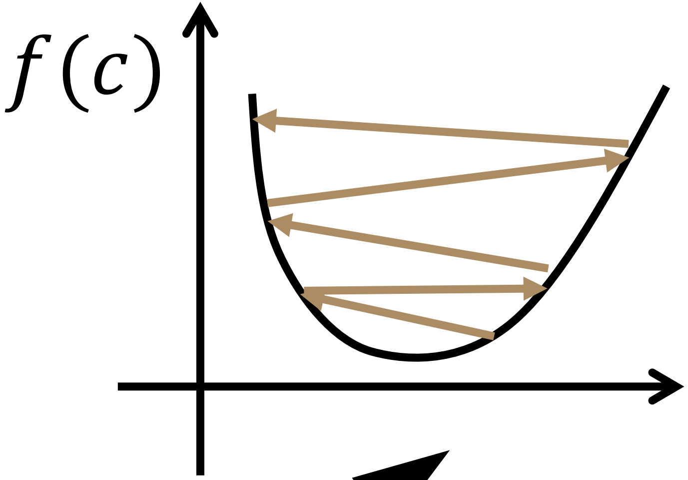




Making Sure Gradient Descent Working Correctly

- Use smaller learning rate α





Very large learning rate

Making Sure Gradient Descent Working Correctly

- Feature scaling:
  - Example: assume features for the house price includes number of bedrooms and living area
  - # of bedrooms between 0 and 5
  - But living area between 1 and 5000 feet^2
  - Make all features have the same level of magnitude

Application for Minimizing Loss Function

- Linear regression: loss function for predictor $p_{\mathbf{c}}(x) = c_0 + c_1 x$ is

$$L(c_0, c_1) = \sum_{i=1}^{M} \left(p\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$= \sum_{i=1}^{M} \left(c_0 + c_1 x^{(i)} - y^{(i)}\right)^2$$

Use gradient descent to $\min\limits_{c_0, c_1} L(c_0, c_1)$

Application for Minimizing Loss Function

- Step 1: Assign initial values for c_0, c_1:
$$c_0 = 0, c_1 = 1$$
- Step 2: Update the change in values for c_0, c_1: $$c_0 := c_0 - \alpha \frac{\partial}{\partial c_0} L(c_0, c_1)$$
$$:= c_0 - \alpha \sum_{i=1}^{M} 2\left(c_0 + c_1 x^{(i)} - y^{(i)}\right)$$

Application for Minimizing Loss Function

- Step 2: (continue)

$$c_1 := c_1 - \alpha \frac{\partial}{\partial c_1} L(c_0, c_1)$$

$$:= c_1 - \alpha \sum_{i=1}^{M} 2x^{(i)} \left(c_0 + c_1 x^{(i)} - y^{(i)}\right)$$

- Step 3: Repeat Step 2 until it converges
- Logistic regression: do it similarly

Stochastic Gradient Descent

- **Stochastic gradient descent (SGD):**
- Herbert Robbins and Sutton Monro (1951)
- Good for large/huge data sets

1) Choose an initial parameter set $\boldsymbol{c}$ and learning rate α
2) Randomly shuffle samples in the training set to update $\boldsymbol{c}$

$$\boldsymbol{c} := \boldsymbol{c} - \alpha \frac{\partial}{\partial \boldsymbol{c}} L\left(\boldsymbol{c}, \mathbf{x}^{(i)}, y^{(i)}\right), i = 1,2,..,M$$

No sum over i

3) Repeat 2) until the convergence is reached.

Stochastic Gradient Descent

SGD with momentum: accelerate SGD

$$\boldsymbol{v} := \gamma \boldsymbol{v} + \alpha \frac{\partial}{\partial \boldsymbol{c}} L\left(\boldsymbol{c}, \mathbf{x}^{(i)}, y^{(i)}\right)$$

$$\boldsymbol{c} := \boldsymbol{c} - \boldsymbol{v}$$

https://distill.pub/2017/momentum/

- Adaptive learning rates are often used.
- If multiple passes are needed, the data can be shuffled for each pass to prevent cycles.

Stochastic Gradient Descent

$$\boldsymbol{g} := \frac{\partial}{\partial \boldsymbol{c}} L\left(c, \mathbf{x}^{(i)}, y^{(i)}\right)$$ (Compute gradient)

$$\boldsymbol{m} := \beta_1 \boldsymbol{m} + (1 - \beta_1)\boldsymbol{g}$$ (Update 1st order momentum)

$$v := \beta_2 v + (1 - \beta_2) g^2$$ (Update 2nd order momentum)

$$\widehat{\boldsymbol{m}} := \frac{\boldsymbol{m}}{\beta_1^k}$$ (Compute corrected-1st order momentum)

$$\hat{v} := \frac{v}{\beta_2^k}$$ (Compute corrected-2nd order momentum)

$$\boldsymbol{c} := \boldsymbol{c} - \alpha \frac{\widehat{\boldsymbol{m}}}{\sqrt{\hat{v}} + \epsilon}$$ (Update parameters)

Adaptive Gradient Descent

- Barzilai-Bowein method (for $L(\boldsymbol{c})$ convex and $\frac{\partial}{\partial \boldsymbol{c}} L(\boldsymbol{c})$ Lipschitz):
- $\boldsymbol{c}^n = \boldsymbol{c}^{n-1} - \alpha^{\boldsymbol{n}} \frac{\partial}{\partial \boldsymbol{c}} L(\boldsymbol{c})$

$$\alpha^{\boldsymbol{n}} = \frac{(\boldsymbol{c}^n - \boldsymbol{c}^{n-1})^T \left[\left.\frac{\partial}{\partial \boldsymbol{c}} L(\boldsymbol{c})\right|_{\boldsymbol{c}=\boldsymbol{c}^n} - \left.\frac{\partial}{\partial \boldsymbol{c}} L(\boldsymbol{c})\right|_{\boldsymbol{c}=\boldsymbol{c}^{n-1}} \right]}{\left\| \left.\frac{\partial}{\partial \boldsymbol{c}} L(\boldsymbol{c})\right|_{\boldsymbol{c}=\boldsymbol{c}^n} - \left.\frac{\partial}{\partial \boldsymbol{c}} L(\boldsymbol{c})\right|_{\boldsymbol{c}=\boldsymbol{c}^{n-1}} \right\|^2}$$

Convex => the global minimum!

Stochastic Gradient Descent

- A Method for Stochastic Optimization (Adam) by Kingma & Ba, 2015: An efficiency version of SGD using first and second order momentum, well suited for large data set problems
- Kalman-based Stochastic Gradient Descent: *SIAM Journal on Optimization*. **26** (4): 2620–2648. arXiv:1512.01139

Discussions

Pros and Cons of Gradient Descent

- **Pros**
  - Can be applied for any dimensional space
  - Nonlinear problems
  - Easy to implement
- **Cons:**
  - Local optima problem
  - Slowly to reach the local minimum
  - Cannot be applied for discontinuous functions

Discussions

- **Sample noise (uncertainty in $\{y^{(i)}\}$)**
- **Parameter linear dependence (in $\{c_i\}$)**
- **Manifold properties:**
  - **Smoothness -- differentiability**
  - **Convex/concave**
  - **Tangent bundle/cotangent bundle**
  - **Topological structure of the tangent space**
  - **de Rham-Hodge theory**

Feature Normalization

- Purpose: all features will have relatively similar magnitude
- How?:
  1. Linearly scale features to range $[0,1]$

$$x_{\text{new}} = \frac{x_{\text{old}} - x_{\text{old}}^{\min}}{x_{\text{old}}^{\max} - x_{\text{old}}^{\min}}$$

  2. Linearly scale features to 0 mean and variance 1 (normal distribution)

$$x_{\text{new}} = \frac{x_{\text{old}} - \mu}{\sigma}$$

$$\mu\text{: mean}, \sigma^2\text{: variance}$$

Feature Normalization

Original dataset

Name	x_1	x_2	Label
P1	1	100	Red
P2	1	120	Red
P3	4	200	Green
P4	4	250	Green

Feature Normalization

After normalizing features using normal distribution
$(\mu(x_1) = 2.5, \sigma(x_1) = 1.5, \mu(x_2) = 167.5, \sigma(x_2) \approx 60.57)$

Name	x_1	x_2	label
P1	-1	-1.11	Red
P2	-1	-0.78	Red
P3	1	0.54	Green
P4	1	1.36	Green

Feature Normalization

Test set (original)

Name	x_1	x_2	label
P5	1	220	?

Test set (after normalization)

Name	x_1	x_2	label
P5	-1	0.87	?

Feature Normalization

Training set (normalized)

Name	x_1	x_2	label
P1	-1	-1.11	Red
P2	-1	-0.78	Red
P3	1	0.54	Green
P4	1	1.36	Green

Test set (normalized)

Name	x_1	x_2	label
P5	-1	0.87	?