Gradient Descent

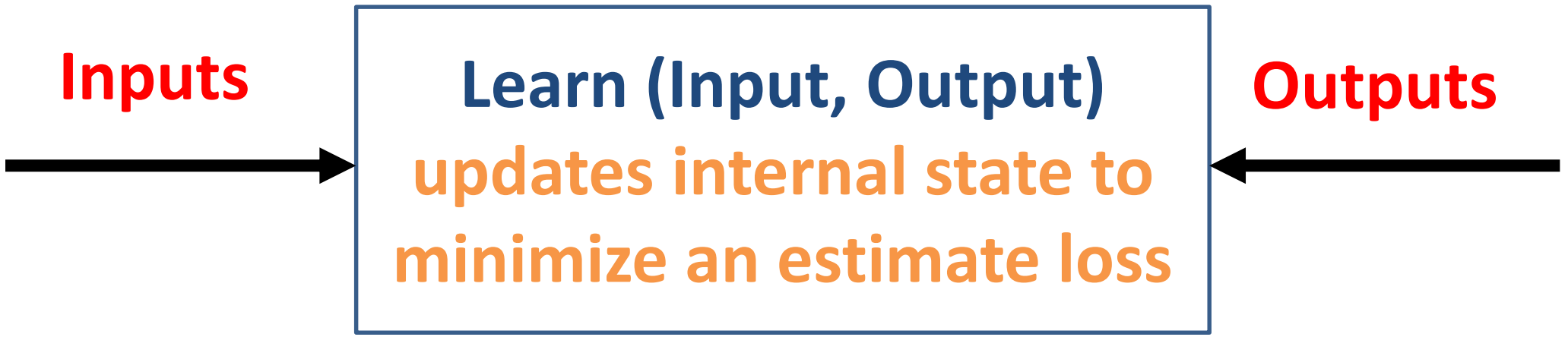
Chul Min Yeum
Assistant Professor
Civil and Environmental Engineering
University of Waterloo, Canada

CIVE 497 – CIVE 700: Smart Structure Technology

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Last updated: 2020-03-27

Supervised Learning

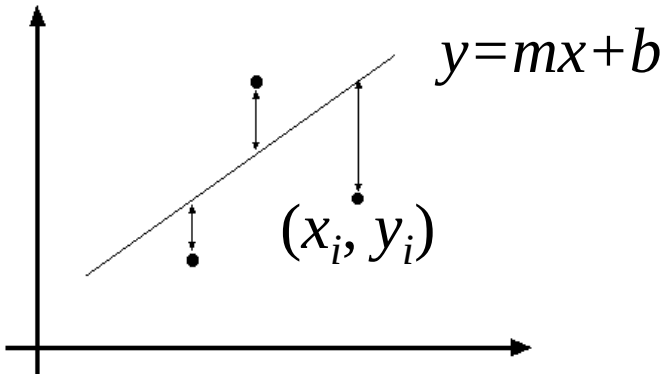


Recall: Least Squares Line Fitting

Data (measurement): $(x_1, y_1), \ldots, (x_n, y_n)$

Model: Line ($y_i = m x_i + b$)

Task: Find (m, b)



Minimize $E = J(m,b) = \sum_{i=1}^{n} (y_i - mx_i - b)^2$

$$\frac{\partial(E)}{\partial m} = -2\sum_{i=1}^{n}[y_i - mx_i - b]x_i = 0$$

$$\frac{\partial(E)}{\partial b} = -2\sum_{i=1}^{n}[y_i - mx_i - b] = 0$$

$$\mathrm{m} = \frac{\sum_{i=1}^{n} x_i y_i - 1/n(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i)}{\sum_{i=1}^{n} x_i^2 - 1/n(\sum_{i=1}^{n} x_i)^2}$$

$$\mathrm{b} = \frac{1/n(\sum_{i=1}^{n} y_i)(\sum_{i=1}^{n} x_i^2) - 1/n \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 - 1/n(\sum_{i=1}^{n} x_i)^2}$$

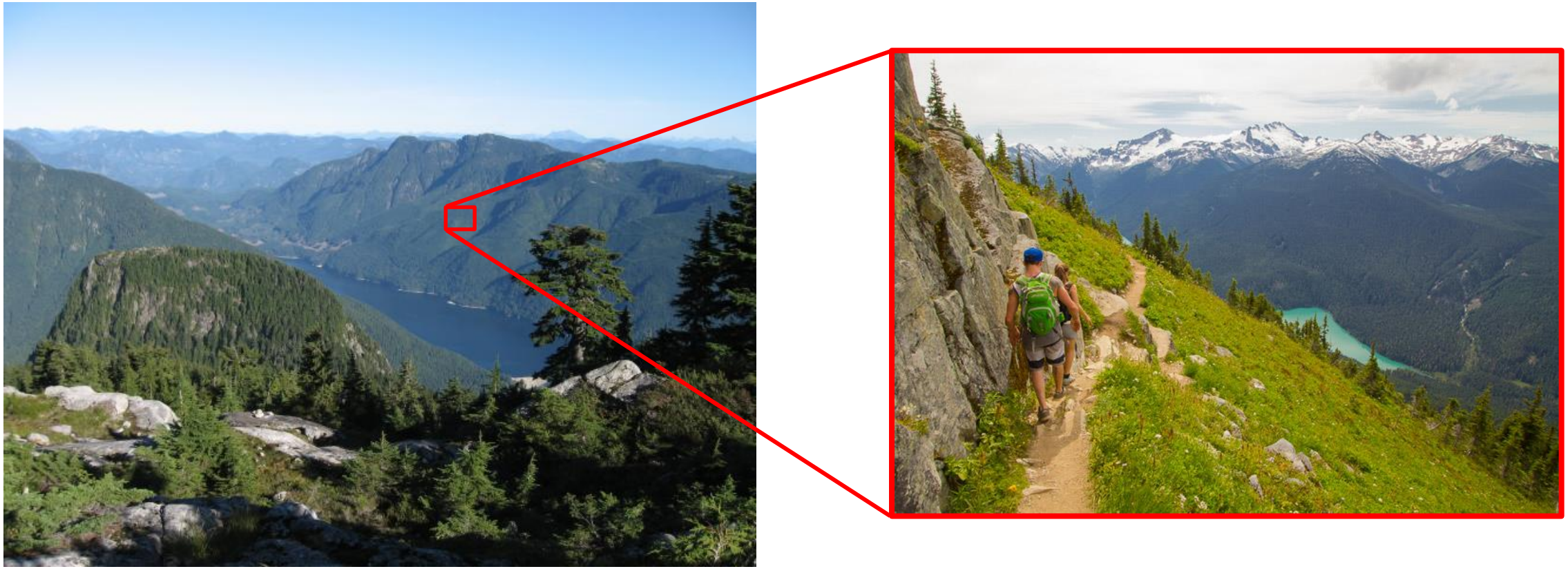
Gradient Descent

- Gradient descent is an iterative machine learning optimization algorithm to find a local minimum of a differentiable function so that we have models that makes accurate predictions.
- Cost function(J) or loss function measures **the difference between the actual output and predicted output** from the model.

Repeat until convergence

$$\theta_{j+1} \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Gradient Descent (Continue)



$$\theta_{j+1} \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Line Fitting using Gradient Descent

Cost function:

$$J(\theta) = J(\theta^1, \theta^2) = \sum_{i=1}^{n} (y_i - \theta^1 x_i - \theta^2)^2$$

Data (measurement): $(x_1, y_1), \ldots, (x_n, y_n)$

Model: Line ($y_i = m x_i + b$)

Task: Find (m, b)

Minimize $E = J(m, b) = \sum_{i=1}^{n} (y_i - m x_i - b)^2$

Derivatives:

$$\frac{\partial J(\theta^1, \theta^2)}{\partial \theta^1} = -2 \sum_{i=1}^{n} [y_i - \theta^1 x_i - \theta^2] x_i$$

$$\frac{\partial J(\theta^1, \theta^2)}{\partial \theta^2} = -2 \sum_{i=1}^{n} [y_i - \theta^1 x_i - \theta^2]$$

Updated rules:

$$\theta_{j+1}^1 \leftarrow \theta_j^1 - \alpha \frac{\partial}{\partial \theta_j^1} J(\theta)$$

$$\theta_{j+1}^2 \leftarrow \theta_j^2 - \alpha \frac{\partial}{\partial \theta_j^2} J(\theta)$$

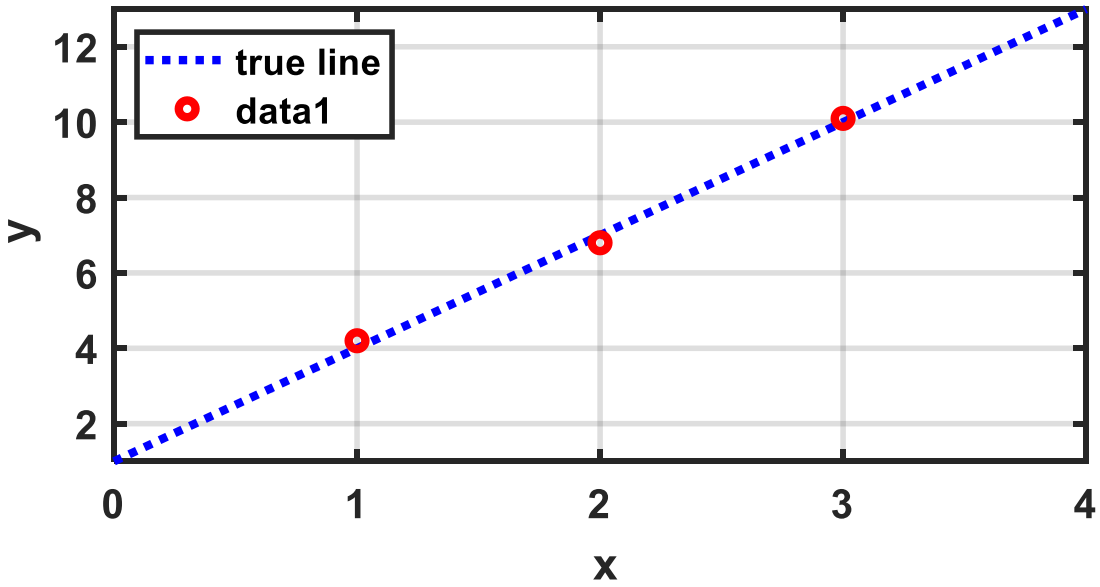
Example: Line Fitting using Gradient Descent

Data (measurement): (1 4.2), (2, 6.8), (3, 10.1)

Model: Line ($y_i = m\,x_i + b$)

True model: y = 3x + 1

Task: Find (m, b)



Least square

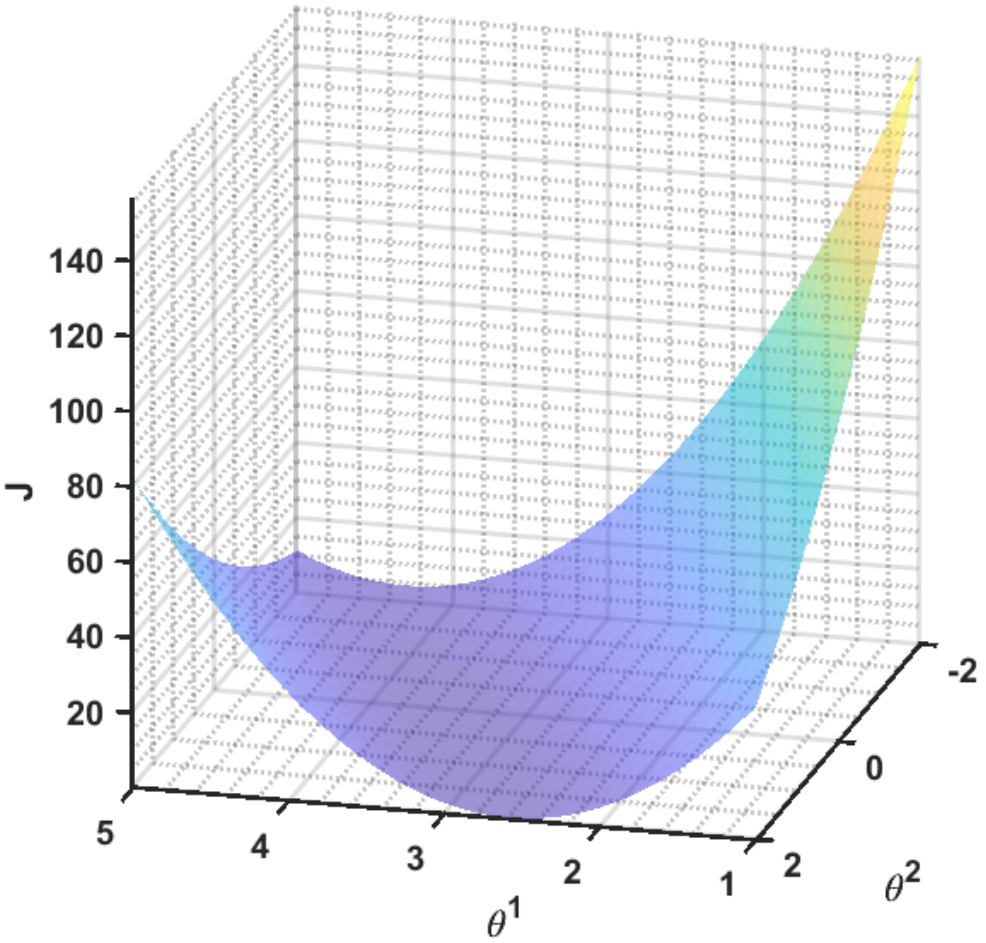
$$\mathrm{m} = \frac{\sum_{i=1}^{n} x_i y_i - 1/n\left(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i\right)}{\sum_{i=1}^{n} x_i^2 - 1/n\left(\sum_{i=1}^{n} x_i\right)^2} = \frac{(4.2 + 13.6 + 30.3) - 1/3(6 * 21.1)}{14 - 1/3(6 * 6)} = \frac{5.9}{2} = 2.95$$

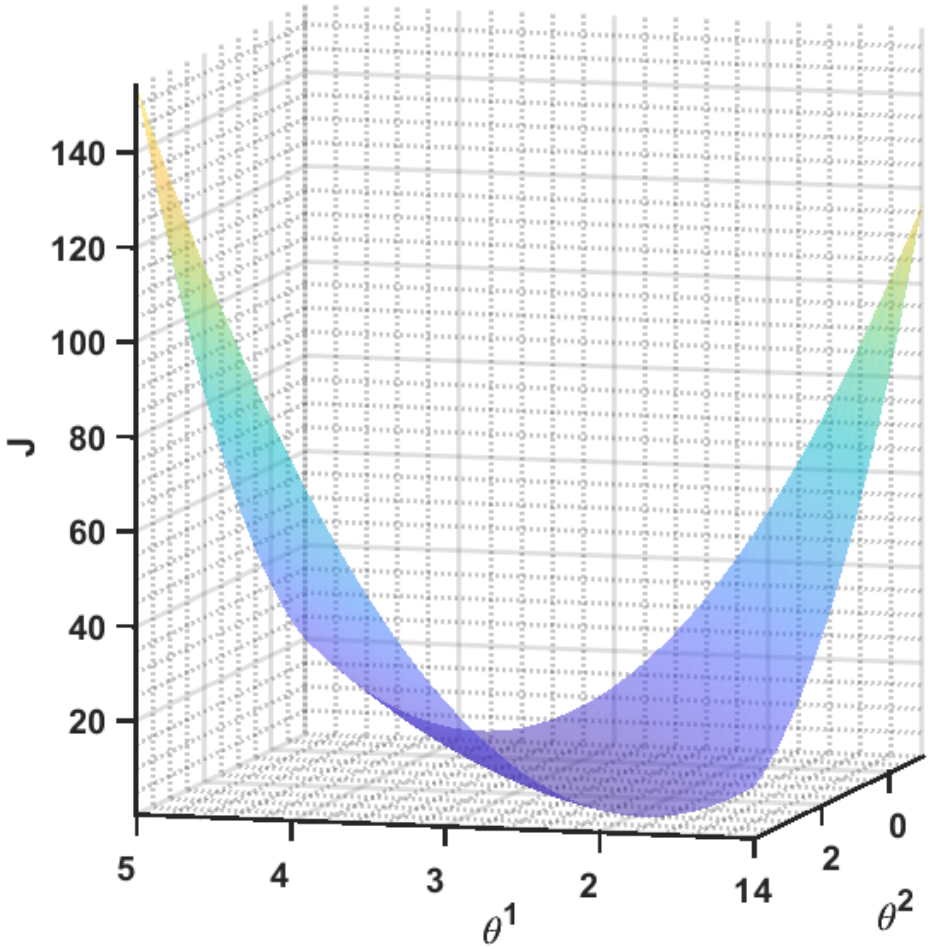
$$\mathrm{b} = \frac{1/n\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right) - 1/n \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 - 1/n\left(\sum_{i=1}^{n} x_i\right)^2} = \frac{\frac{1}{3} * 21.1 * 14 - \frac{1}{3} * 6 * (4.2 + 13.6 + 30.3)}{2} = 1.133$$

Example: Line Fitting using Gradient Descent (Continue)

Cost function: $J(\theta) = J(\theta^1, \theta^2) = \sum_{i=1}^{n} (y_i - \theta^1 x_i - \theta^2)^2$

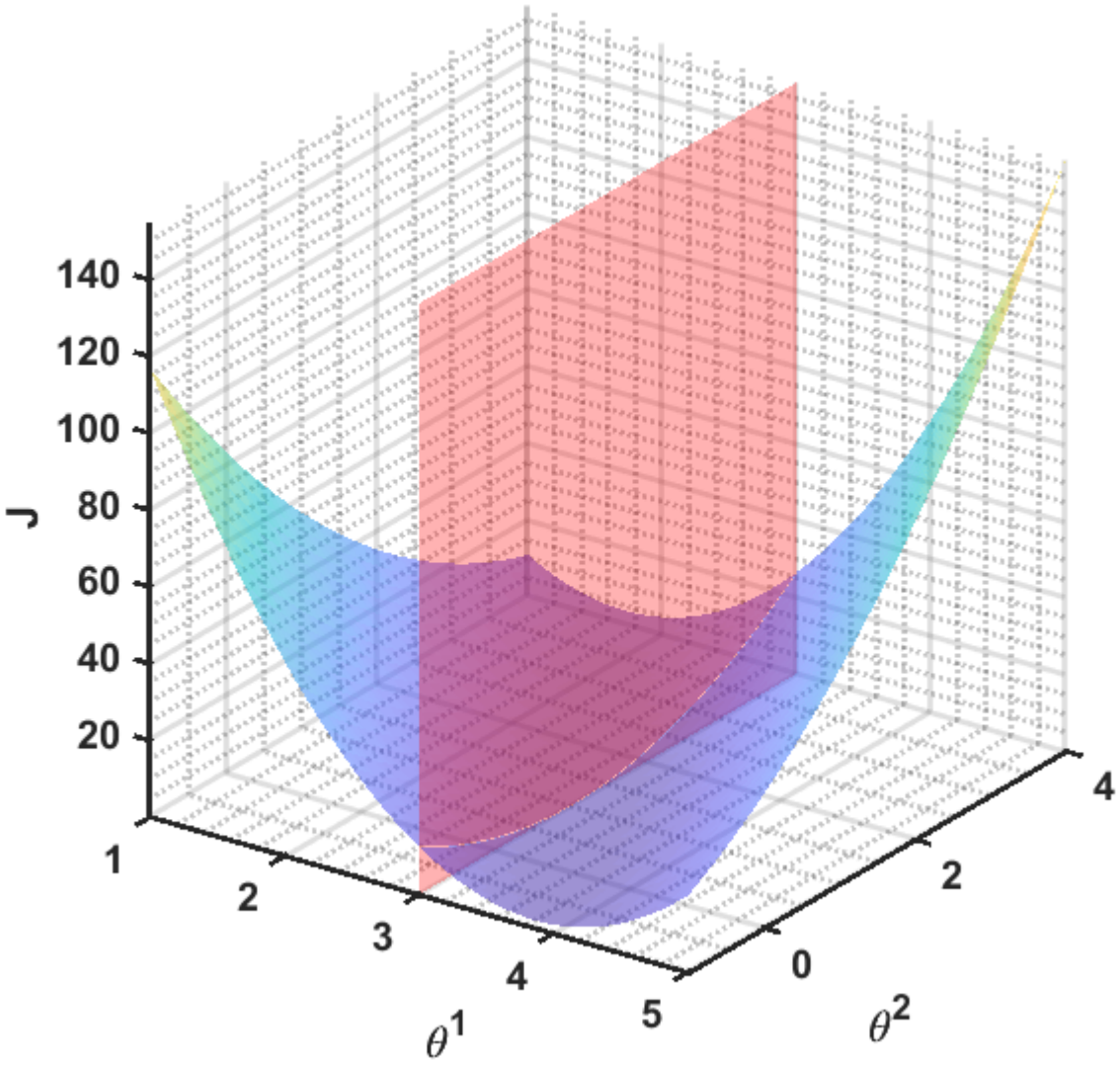
See *demo_line_fit.m*

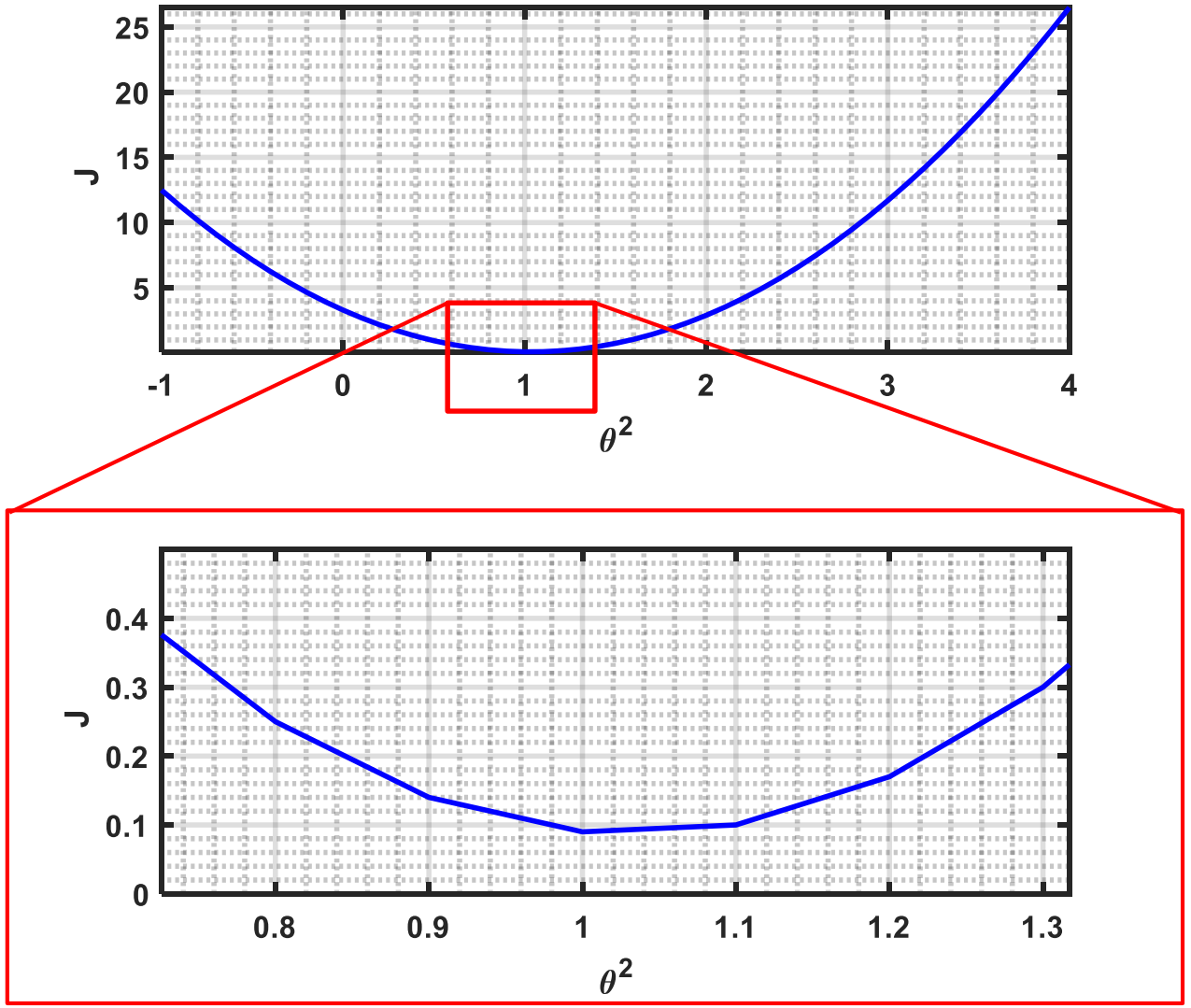




Example: Line Fitting using Gradient Descent (Continue)

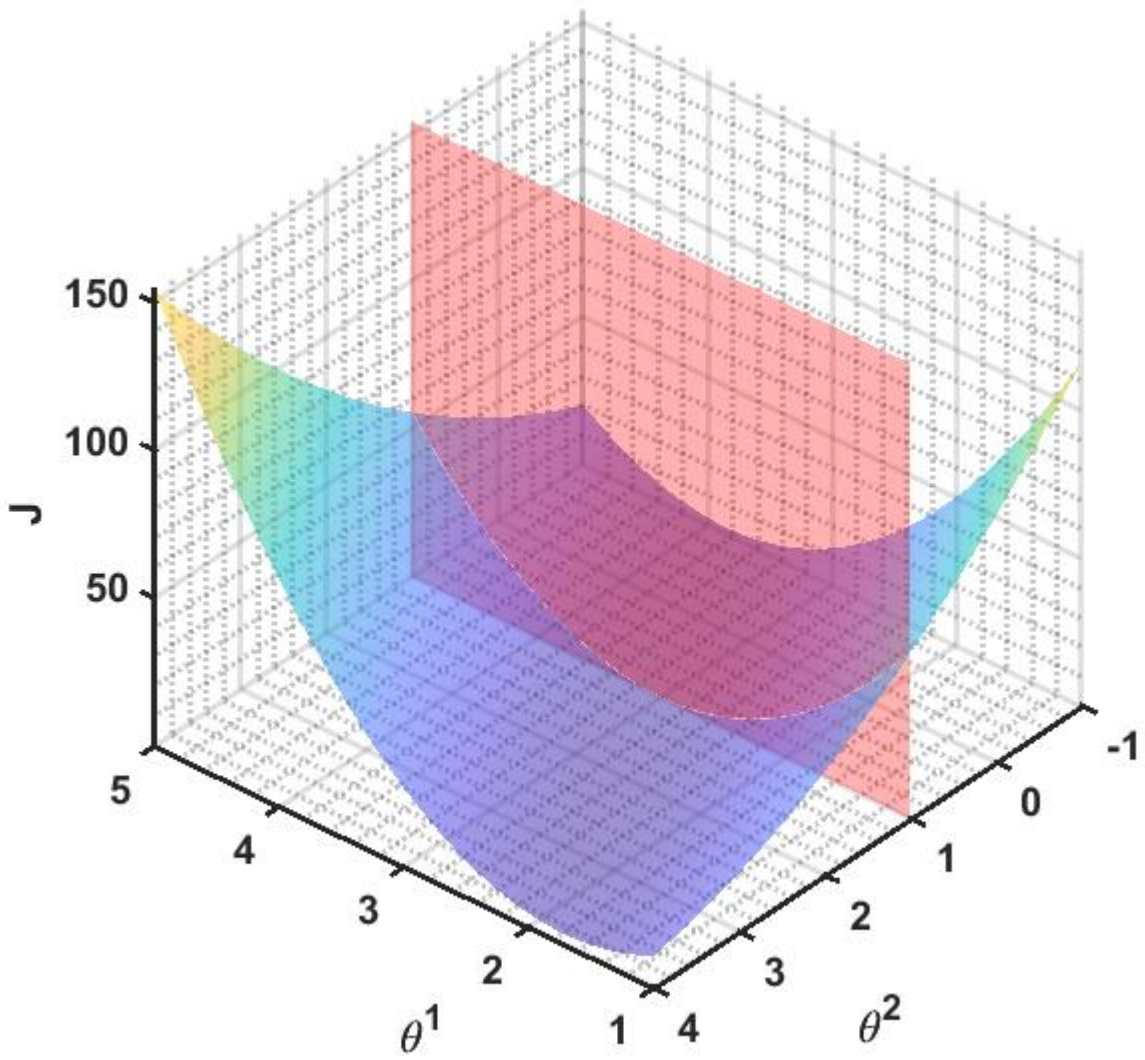
$y_i = 3x_i + 1 = \theta^1 x_i + \theta^2$

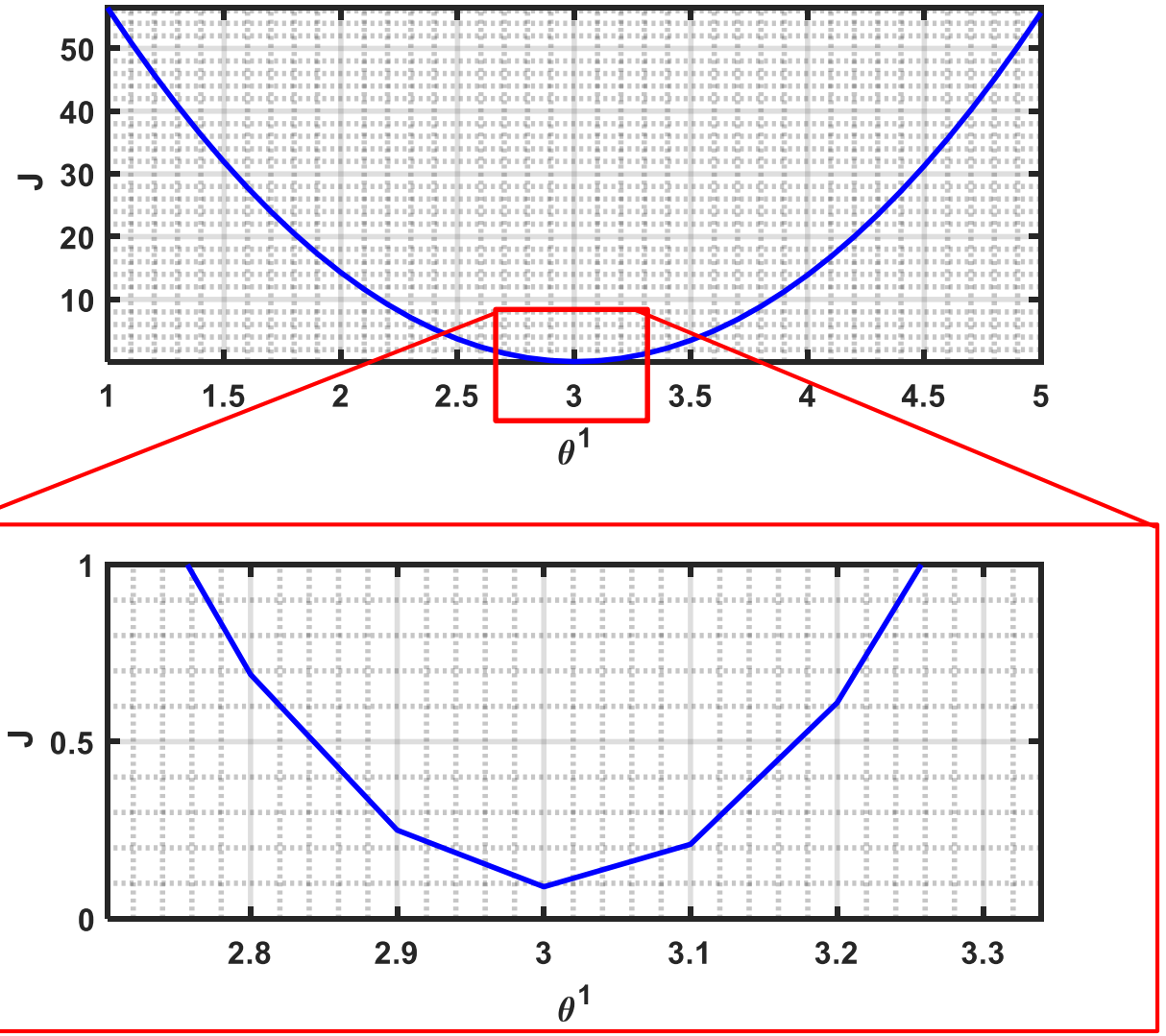




See *demo_line_fit.m*

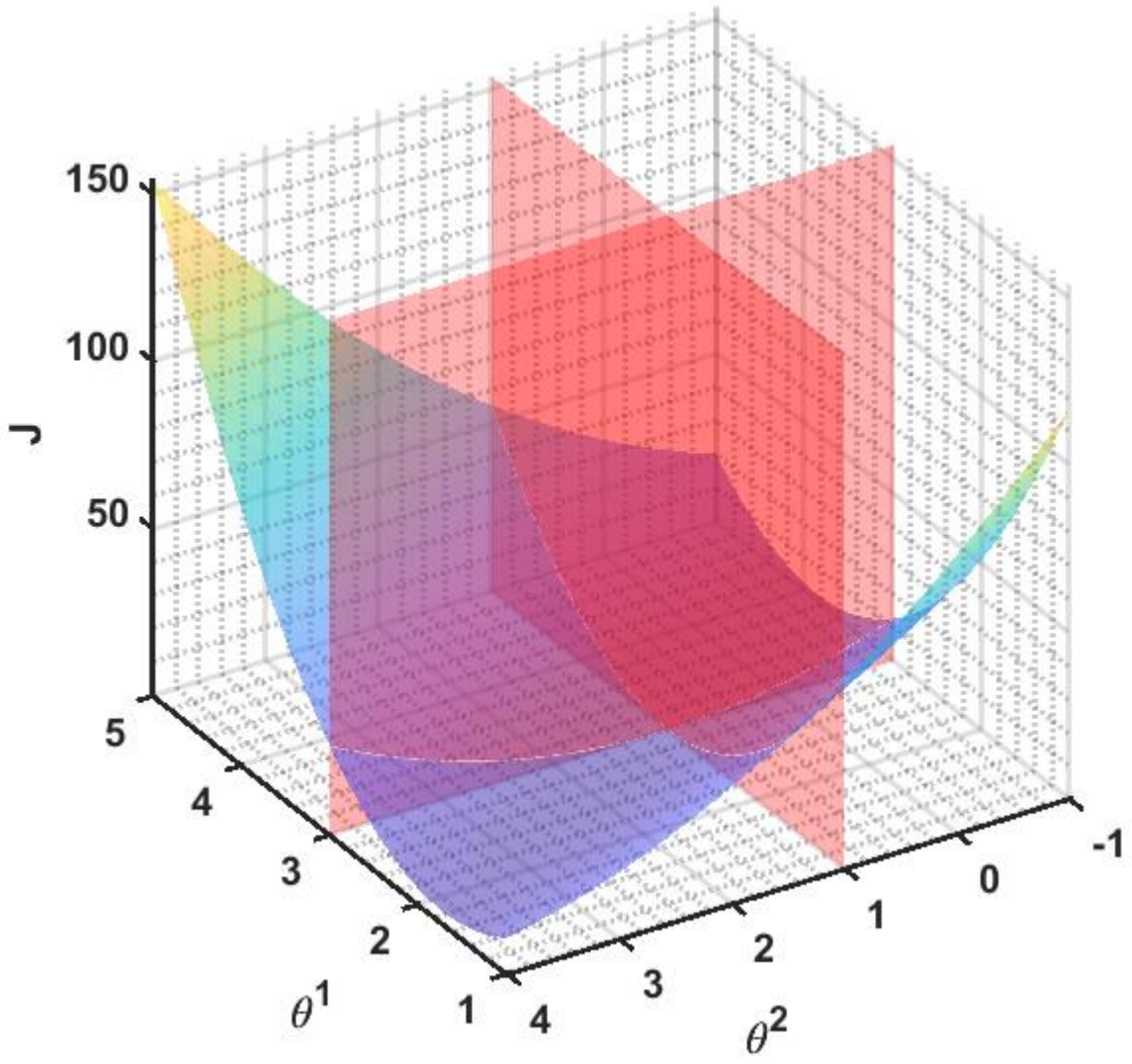
Example: Line Fitting using Gradient Descent (Continue)

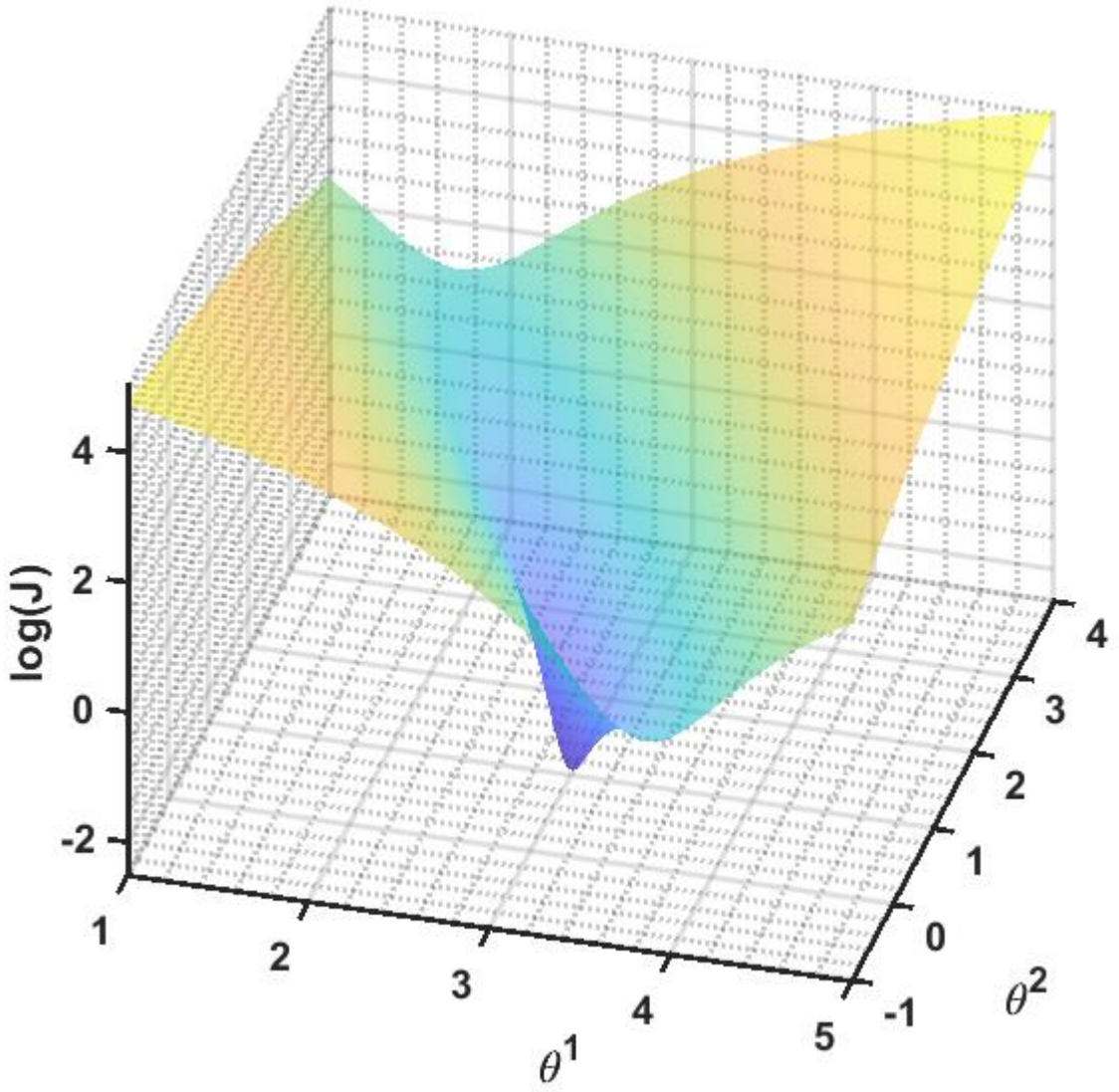




See *demo_line_fit.m*

Example: Line Fitting using Gradient Descent (Continue)





Example: Line Fitting using Gradient Descent (Continue)

$$\frac{\partial J(\theta^1,\theta^2)}{\partial \theta^1} = -2\sum_{i=1}^{n}[y_i - \theta^1 x_i - \theta^2]x_i$$

$$\theta_{j+1}^1 \leftarrow \theta_j^1 - \alpha \frac{\partial}{\partial \theta_j^1} J(\theta)$$

$$\frac{\partial J(\theta^1,\theta^2)}{\partial \theta^2} = -2\sum_{i=1}^{n}[y_i - \theta^1 x_i - \theta^2]$$

$$\theta_{j+1}^2 \leftarrow \theta_j^2 - \alpha \frac{\partial}{\partial \theta_j^2} J(\theta)$$

```
1  in_data = [1 2 3];
2  out_data = [4.2 6.8 10.1];
3
4  itr = 12;
5  l_r = 0.08;
6
7  t1 = zeros(itr,1); t1(1) = 1;
8  t2 = zeros(itr,1); t2(1) = 0;
```

```
9  J1 = @(th1, t2) -2*sum((out_data-t1*in_data-t2).*in_data);
10 J2 = @(th1, t2) -2*sum((out_data-t1*in_data-t2));
11
12 for ii=1:n_iter-1
13     t1(ii+1) = t1(ii) - l_r*J1(t1(ii), t2(ii));
14     t2(ii+1) = t2(ii) - l_r*J2(t1(ii), t2(ii));
15 end
16
```

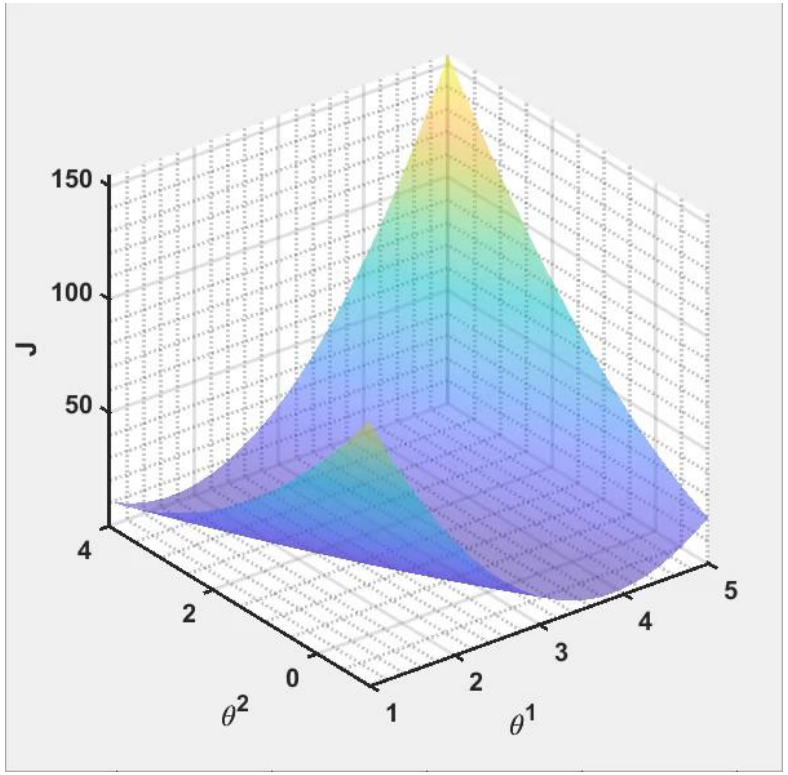
Line Fitting using Gradient Descent Depending on Learning Rates

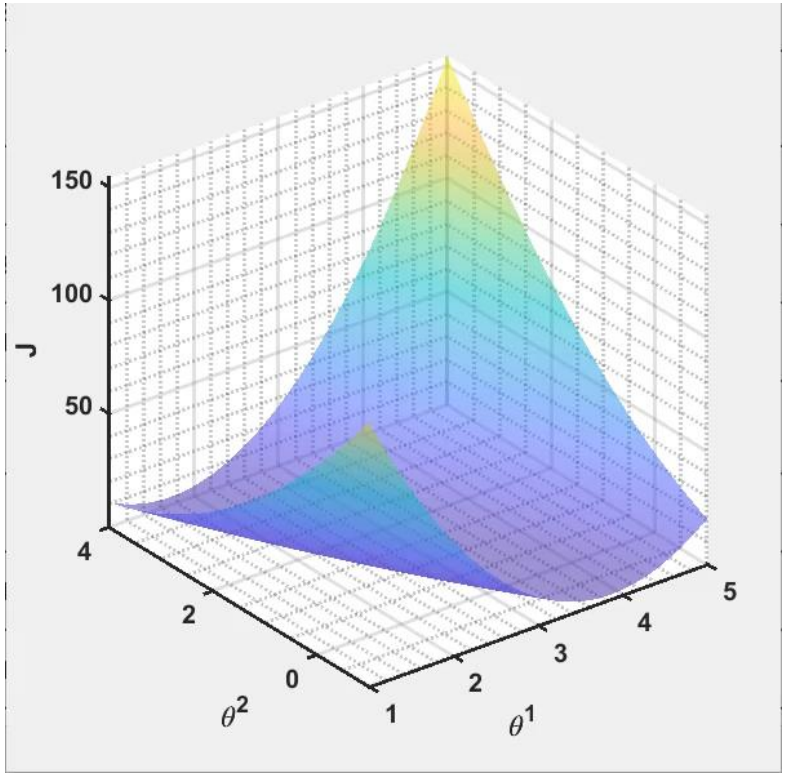
$\alpha = 0.01$

Iter.	1	2	3	4	5	6	7	8	9	10	11	12
θ^1	1	1.682	2.1368	2.440011	2.642082	2.776673	2.866242	2.925774	2.965265	2.991388	3.008593	3.019848
θ^2	0	0.302	0.50404	0.639382	0.730217	0.791354	0.832672	0.860763	0.880024	0.893391	0.902821	0.909621

$\alpha = 0.001$

Iter.	1	2	3	4	5	6	7	8	9	10	11	12
θ^1	1	1.0682	1.134128	1.19786	1.259468	1.319024	1.376595	1.432248	1.486047	1.538053	1.588325	1.636923
θ^2	0	0.0302	0.0594	0.087634	0.114934	0.141331	0.166855	0.191535	0.215398	0.238473	0.260786	0.282361





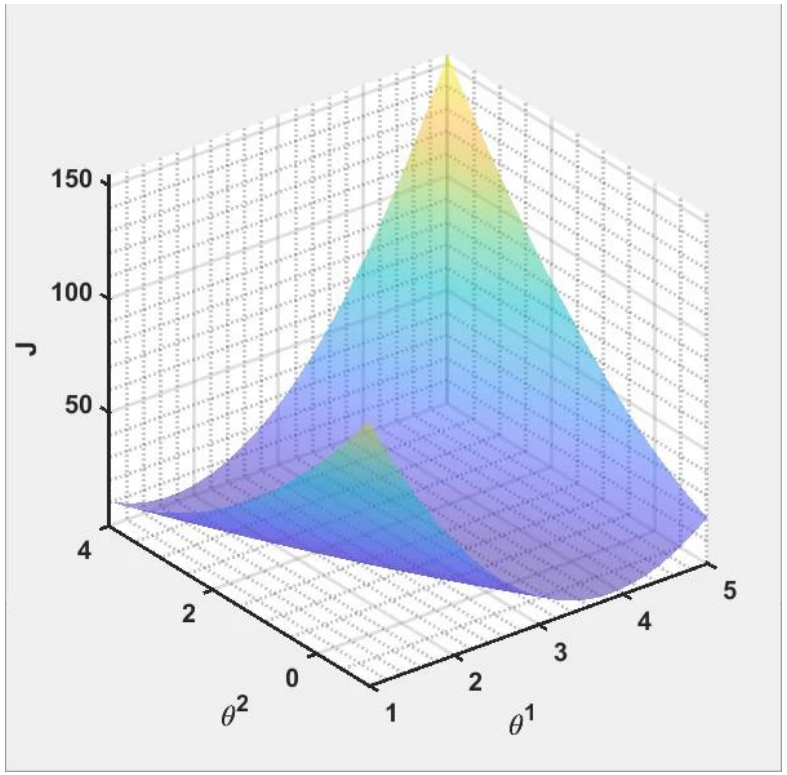
Line Fitting using Gradient Descent Depending on Learning Rates (Continue)

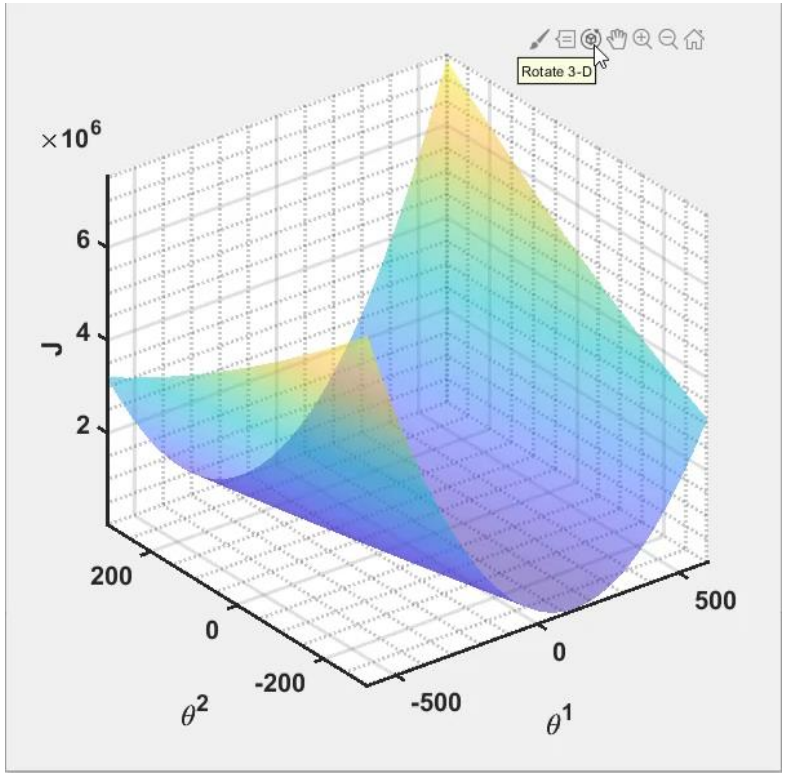
$\alpha = 0.05$

Iter.	1	2	3	4	5	6	7	8	9	10	11	12
θ^1	1	4.41	2.14	3.6414	2.63902	3.299202	2.855733	3.145209	2.948232	3.074404	2.98619	3.040475
θ^2	0	1.51	0.521	1.1907	0.75865	1.057643	0.870829	1.00614	0.927173	0.990081	0.958415	0.989177

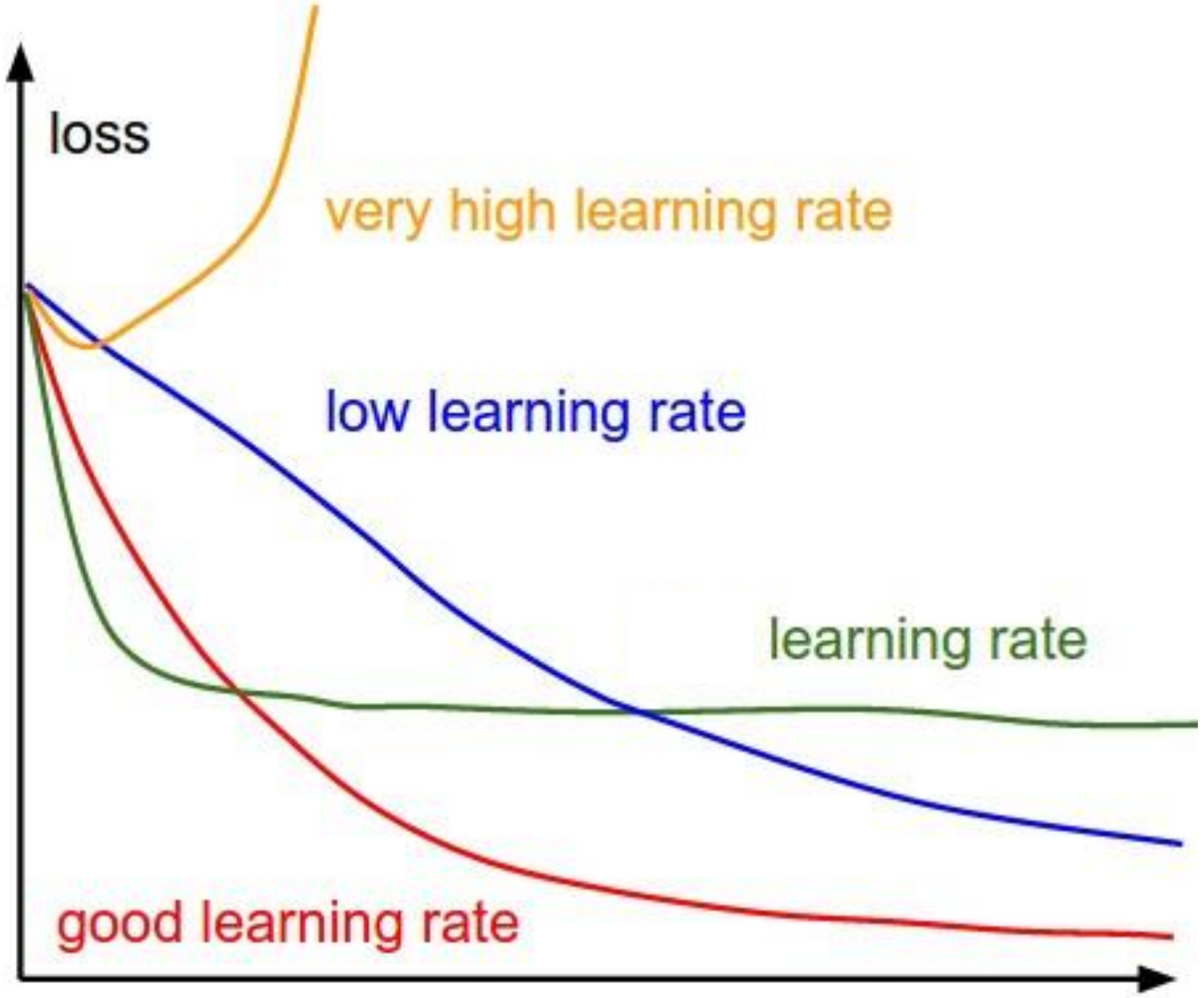
$\alpha = 0.08$

Iter.	1	2	3	4	5	6	7	8	9	10	11	12
θ^1	1	6.456	-2.6288	12.45853	-12.6348	29.06521	-40.2649	74.97162	-116.597	201.8388	-327.509	552.42
θ^2	0	2.416	-1.56544	5.085619	-5.93967	12.41676	-18.0699	32.63401	-51.6271	88.46348	-144.388	242.7028



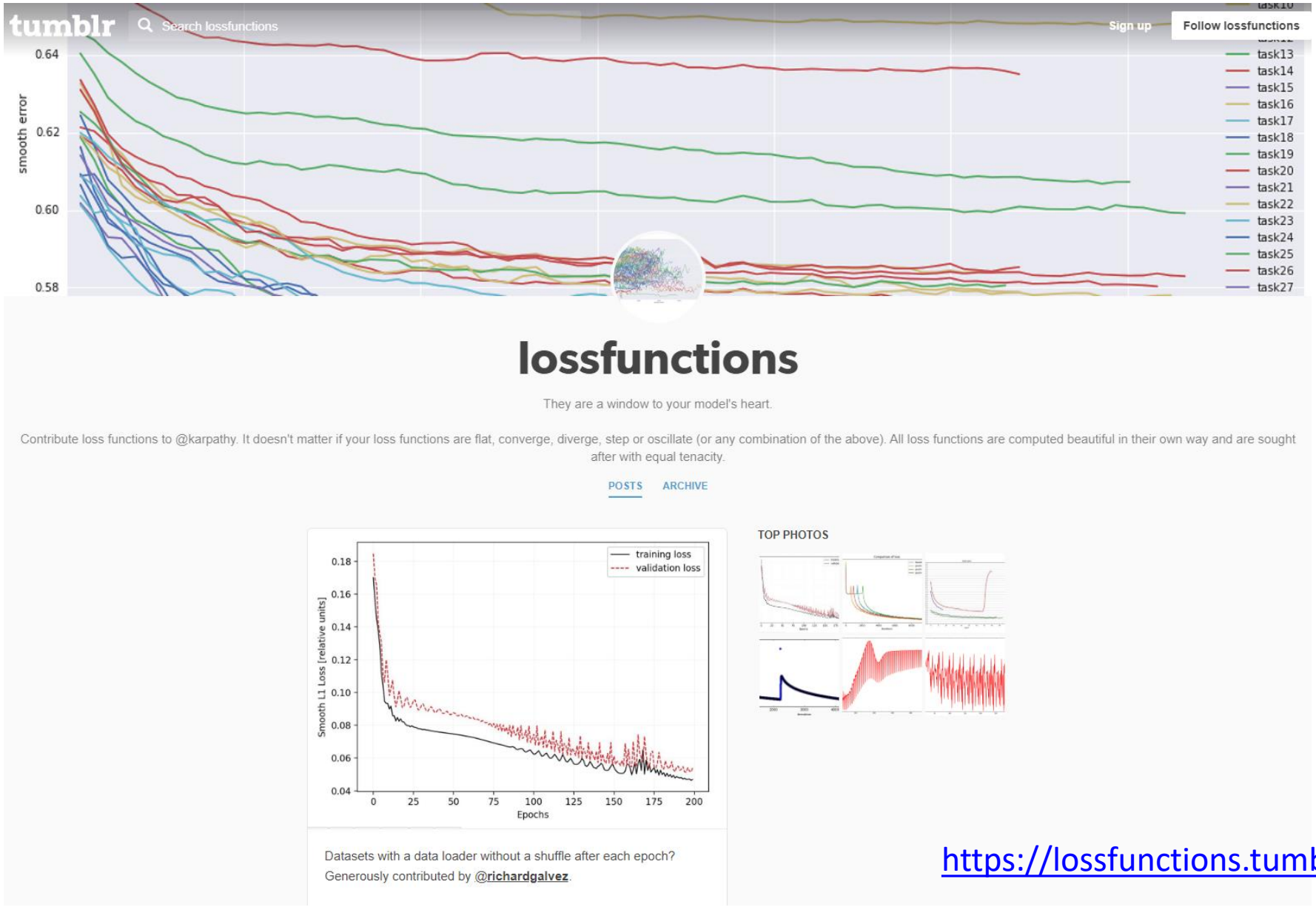


Effect of Learning Rates



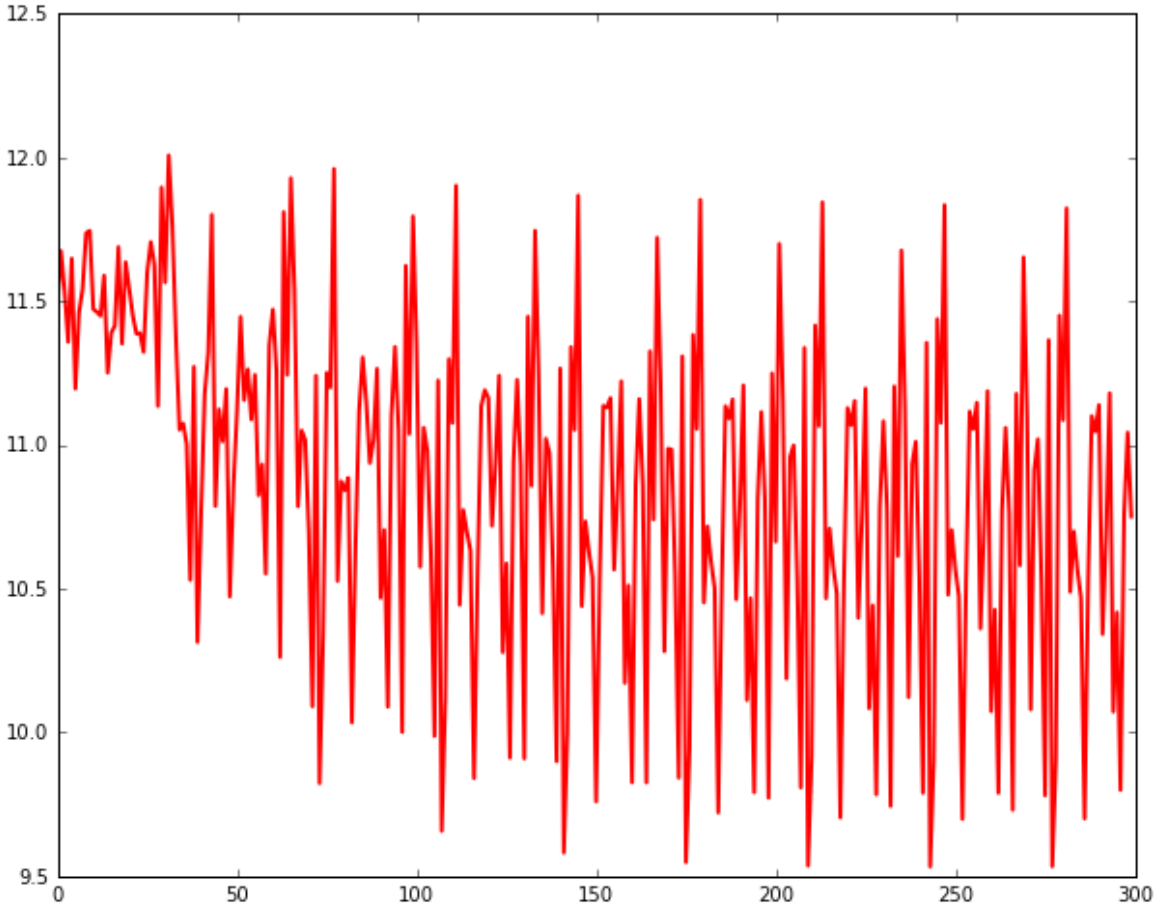
Higher or lower

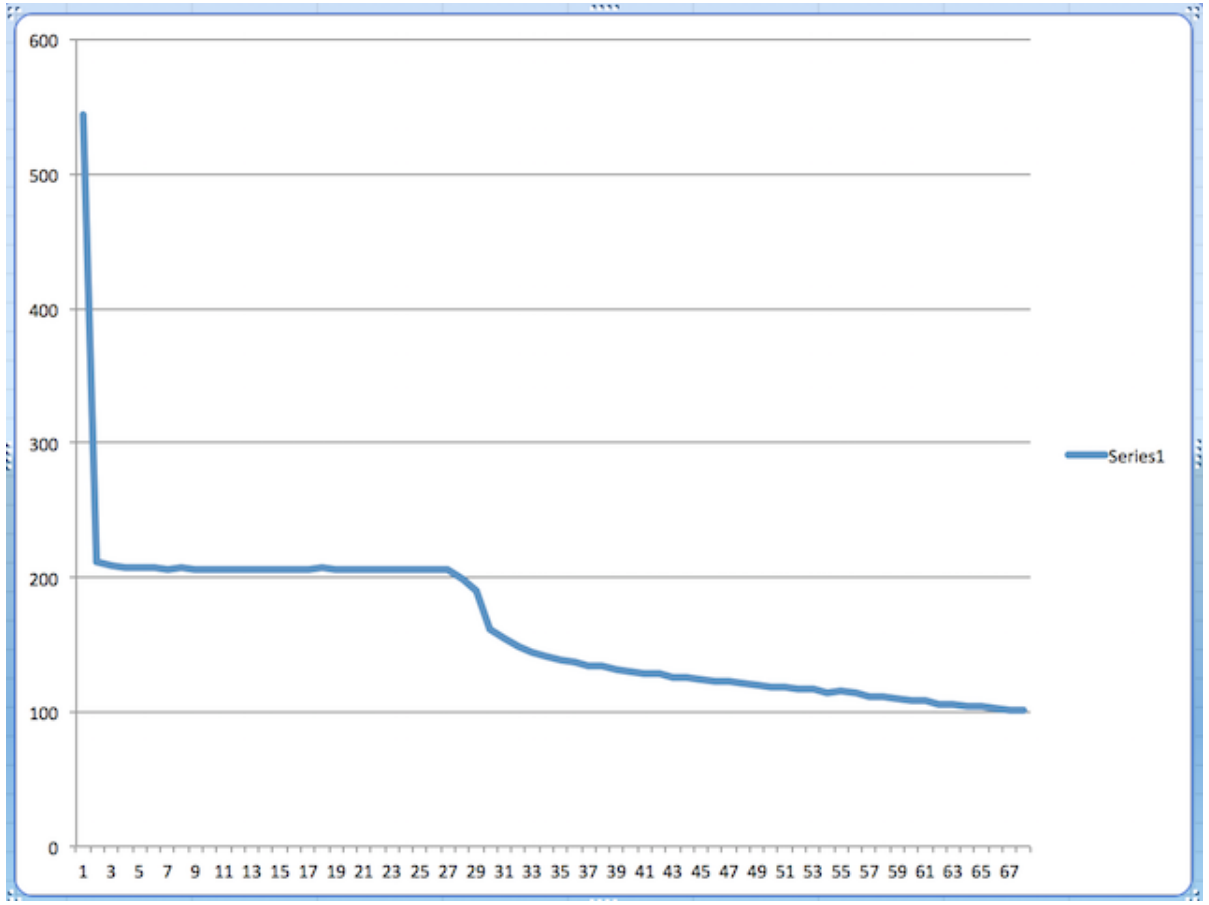
Loss Functions



https://lossfunctions.tumblr.com/

Samples of Loss Curves





Stochastic Gradient Decent

Batch (Vanilla) Gradient Descent

$$\theta_{j+1} \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

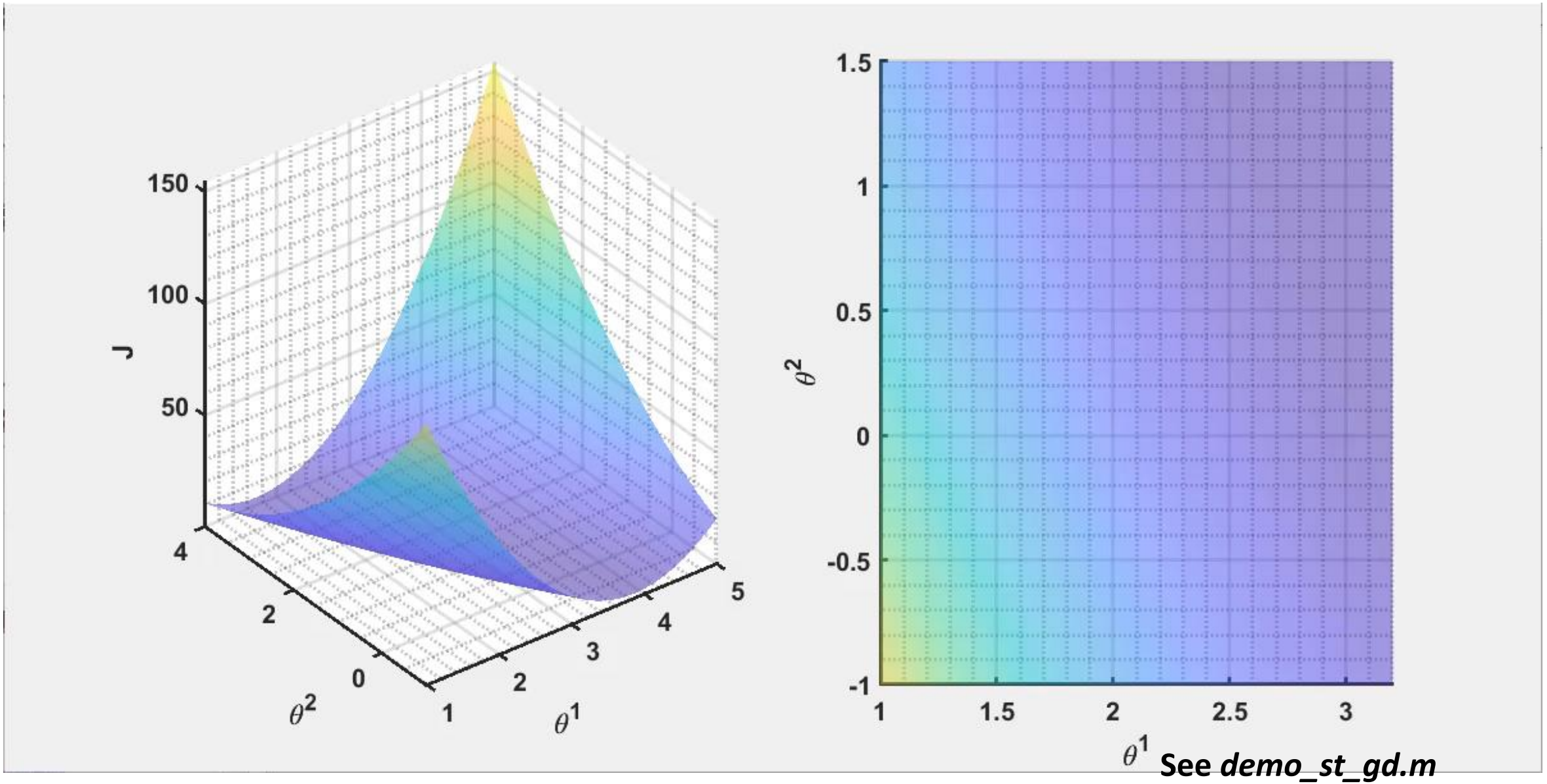
Stochastic Gradient Descent

$$\theta_{j+1} \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(x^i, y^j; \theta)$$

Mini-batch Gradient Descent

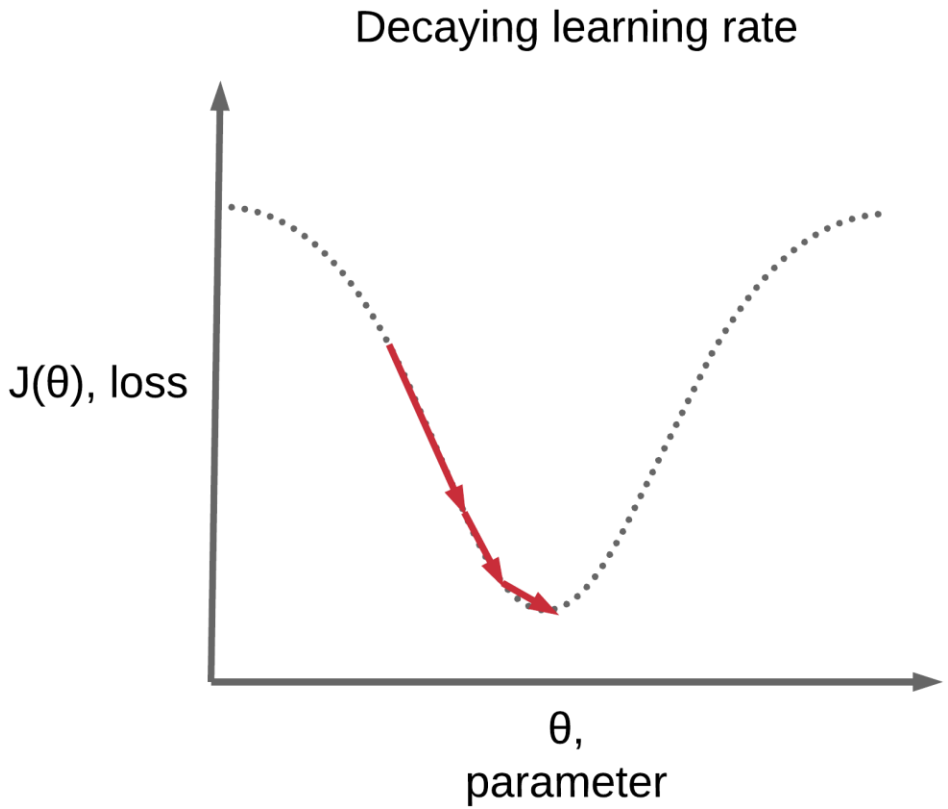
$$\theta_{j+1} \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(x^{\{i:i+b\}}, y^{\{i:i+b\}}; \theta)$$

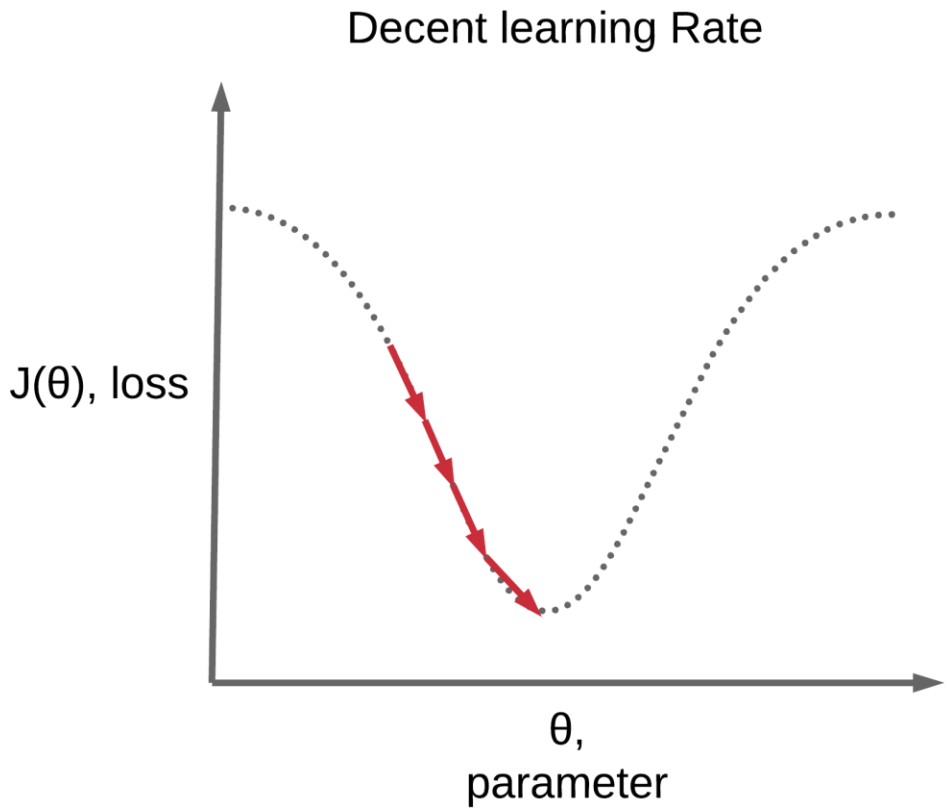
Stochastic Gradient Decent (Simulation)



See *demo_st_gd.m*

Variations (Adaptive Learning Rate or Learning Rate Scheduling)





$$\theta_{j+1} \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\alpha_{\mathrm{j}} = \alpha_{\mathrm{j}} \gamma \quad \text{where } \gamma = 0.1$$

Variations (Momentum)

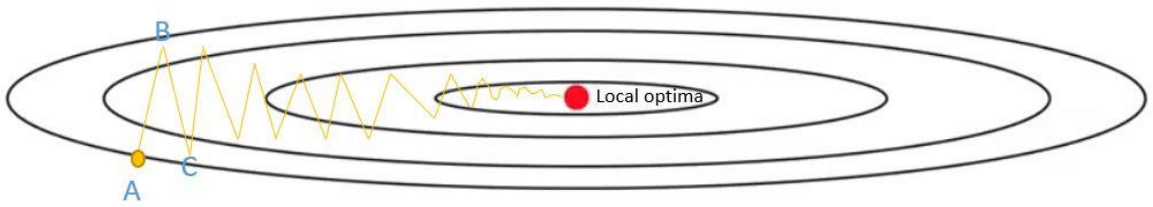
Without momentum:

$$w_{t+1} = w_t - \eta \nabla w_t$$

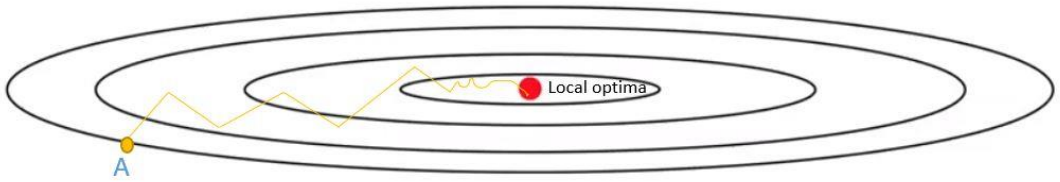
With momentum:

$$update_t^w = \gamma \cdot update_{t-1}^w + \eta \nabla w_t$$
$$w_{t+1} = w_t - update_t^w$$

$$\begin{aligned}
update_0 &= 0 \\
update_1 &= \gamma \cdot update_0 + \eta \nabla w_1 = \eta \nabla w_1 \\
update_2 &= \gamma \cdot update_1 + \eta \nabla w_2 = \gamma \cdot \eta \nabla w_1 + \eta \nabla w_2 \\
update_3 &= \gamma \cdot update_2 + \eta \nabla w_3 = \gamma(\gamma \cdot \eta \nabla w_1 + \eta \nabla w_2) + \eta \nabla w_3 \\
&= \gamma \cdot update_2 + \eta \nabla w_3 = \gamma^2 \cdot \eta \nabla w_1 + \gamma \cdot \eta \nabla w_2 + \eta \nabla w_3 \\
update_4 &= \gamma \cdot update_3 + \eta \nabla w_4 = \gamma^3 \cdot \eta \nabla w_1 + \gamma^2 \cdot \eta \nabla w_2 + \gamma \cdot \eta \nabla w_3 + \eta \nabla w_4 \\
&\vdots \\
update_t &= \gamma \cdot update_{t-1} + \eta \nabla w_t = \gamma^{t-1} \cdot \eta \nabla w_1 + \gamma^{t-2} \cdot \eta \nabla w_1 + \ldots + \eta \nabla w_t
\end{aligned}$$



Without momentum



With momentum

https://towardsdatascience.com/learning-parameters-part-2-a190bef2d12

$$\theta_{j+1} \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$