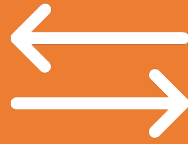


Epoch, Iteration, and Batch size



1 epoch is to pass an entire dataset through the neural network once.



Iteration is the number of mini-batch.



Batch size is the total number of training samples present in a single min-batch.

The background of the slide features a dark blue and black abstract design. On the left side, there is a line graph with several data points connected by white lines. One data point is highlighted with a yellow circle. In the center, there is a large, semi-transparent blue rectangular area containing the number '289.33' in white. The overall aesthetic is modern and technical, suggesting a focus on data analysis or machine learning.

Examples

- We can divide the dataset of 10,000 examples into mini batches of 500 then it will take 20 iterations to complete 1 epoch.
- The batch size is usually 2^2 , 2^3 , 2^4 , 2^5 [...] based on GPU memory.

Gradient Descent

- Gradient descent is an optimization algorithm used to minimize some functions by moving in a certain direction.
- Gradient descent in machine/deep learning is used to update the parameters of models. (weights in neural networks)
- Gradient descent is very useful for supervised learning.



Yann LeCun

@ylecun

I've been trying to convince many of my more theory-oriented colleagues of the unbelievable power of gradient descent for close to 4 decades.

1/2



Chris Olah @ch402 · Jun 4

The elegance of ML is the elegance of biology, not the elegance of math or physics.

Simple gradient descent creates mind-boggling structure and behavior, just as evolution creates the awe inspiring complexity of nature.

twitter.com/banburismus_/s...

[Show this thread](#)

9:11 AM · Jun 5, 2022 · Twitter for Android

x	y
2	4
3	6
4	8
5	10
6	12

$$y = x * 2$$

x	y
2	6
3	9
4	12
5	15
6	18

$$y = x * 3$$

x	y
2	-1.5
3	-1
4	-0.5
5	0
6	0.5

Gradient Descent helps
you find these parameters.


$$y = x * 0.5 - 2.5$$

Weight Bias

x	y
2	8
3	11
4	14
5	17
6	20

$$y = x * 3 + 2$$

Binary Classification



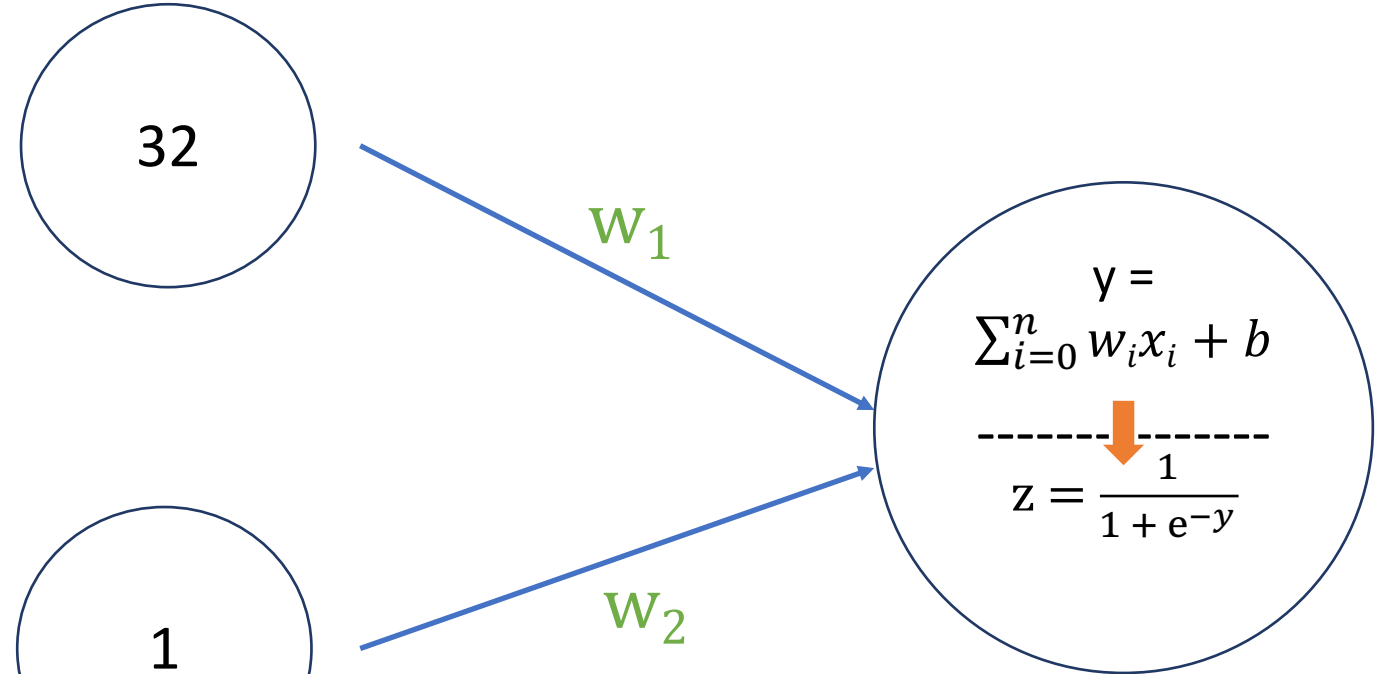
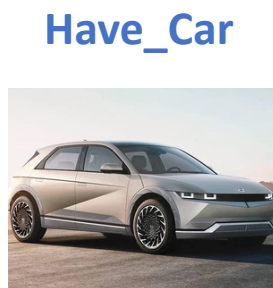
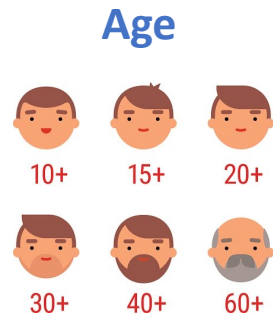
X		y
Age	Have_Car	Have_Insurance
21	0	0
48	1	1
28	0	0
19	0	0
56	1	1
65	1	1
32	1	0
24	0	1
43	1	0
22	0	0
53	1	1

$$y = f(x)$$

Binary Cross Entropy (Log Loss) =

$$-\frac{1}{n} \sum_{i=0}^n y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - (\hat{y}_i))$$

Age	Have_Car	Have_Insurance
21	0	0
48	1	1
28	0	0
19	0	0
56	1	1
65	1	1
32	1	0
24	0	1
43	1	0
22	0	0
53	1	1



After the first epoch: total log loss = 3.32

$$y = w_1 * X_1 + w_2 * X_2 + \text{bias}$$

$$w_1 = w_1 - \text{learning rate} * \partial / \partial w_1$$

$$w_2 = w_2 - \text{learning rate} * \partial / \partial w_2$$

$$w_3 = w_3 - \text{learning rate} * \partial / \partial w_3$$

$$b = b - \text{learning rate} * \partial / \partial b$$

Learning rate is normally 0.01.

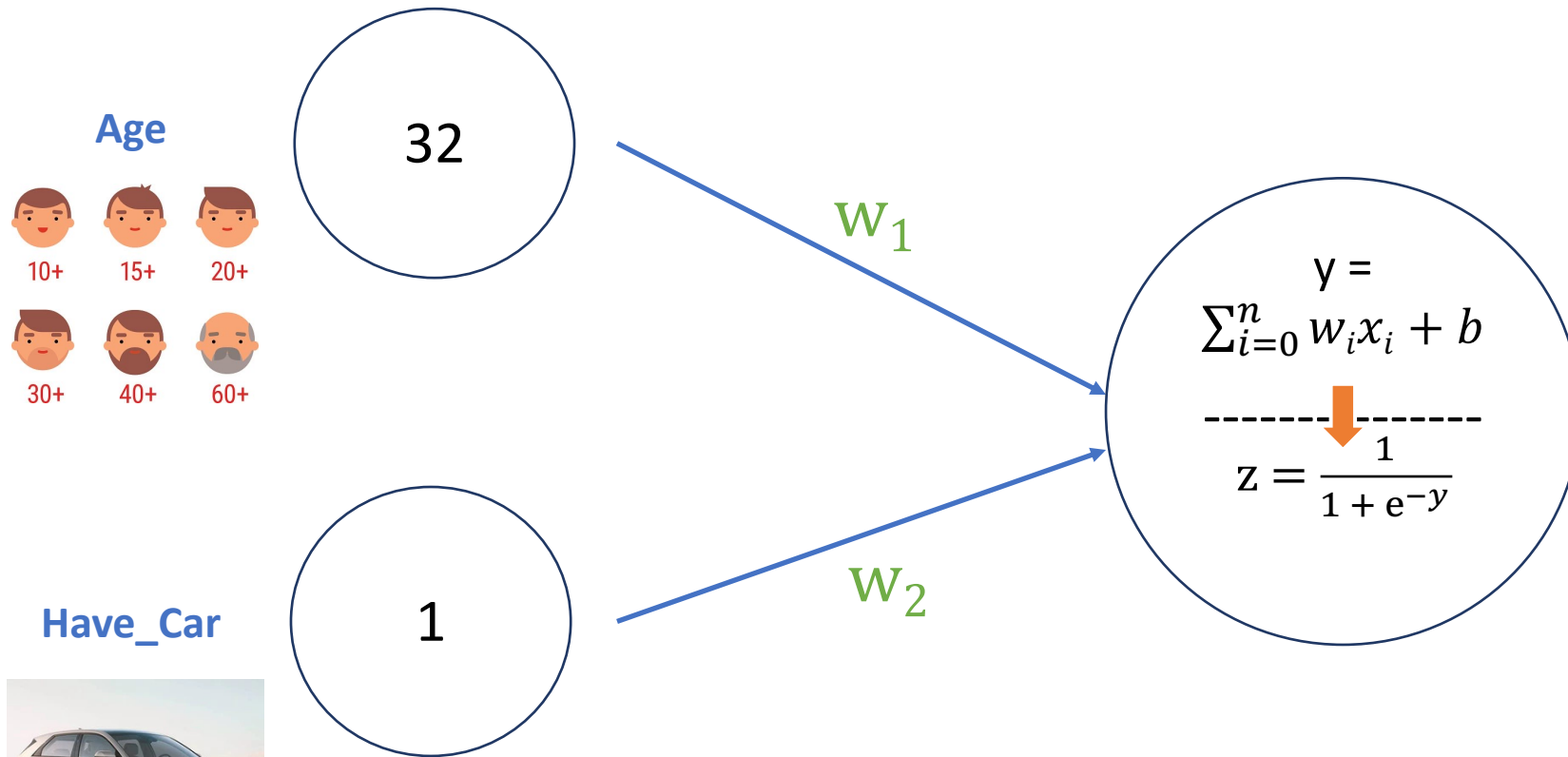
$$w_1 = w_1 - \text{learning rate} * \partial / \partial w_1$$

$$\partial / \partial w_1 = \frac{1}{n} \sum_{i=1}^n x_i (\hat{y}_i - y_i)$$

$$b = b - \text{learning rate} * \partial / \partial b$$

$$\partial / \partial b = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)$$

epoch = 1



$$y = w_1 * X_1 + w_2 * X_2 + \text{bias}$$

$$y = 1 * X_1 + 1 * X_2 + 0$$

$$w_1 = w_1 - \text{learning rate} * \partial / \partial w_1$$

$$w_1 = 1 - 0.2 = 0.8$$

$$w_2 = w_2 - \text{learning rate} * \partial / \partial w_2$$

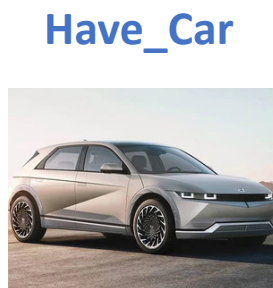
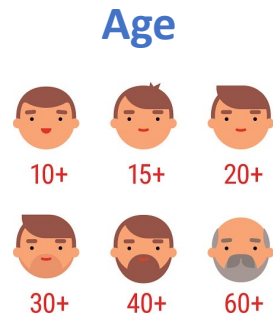
$$w_2 = 0.9 - 0.15 = 0.75$$

$$b = b - \text{learning rate} * \partial / \partial b$$

$$b = 0 - 0.2 = -0.2$$

$$MAE = \frac{1}{n} \sum_{i=0}^n abs(y_i - \hat{y}_i)$$

Age	Have_Car	Have_Insurance
21	0	0
48	1	1
28	0	0
19	0	0
56	1	1
65	1	1
32	1	0
24	0	1
43	1	0
22	0	0
53	1	1



32

1

$$y = w_1 * X_1 + w_2 * X_2 + \text{bias}$$

$$w_1 = 1$$

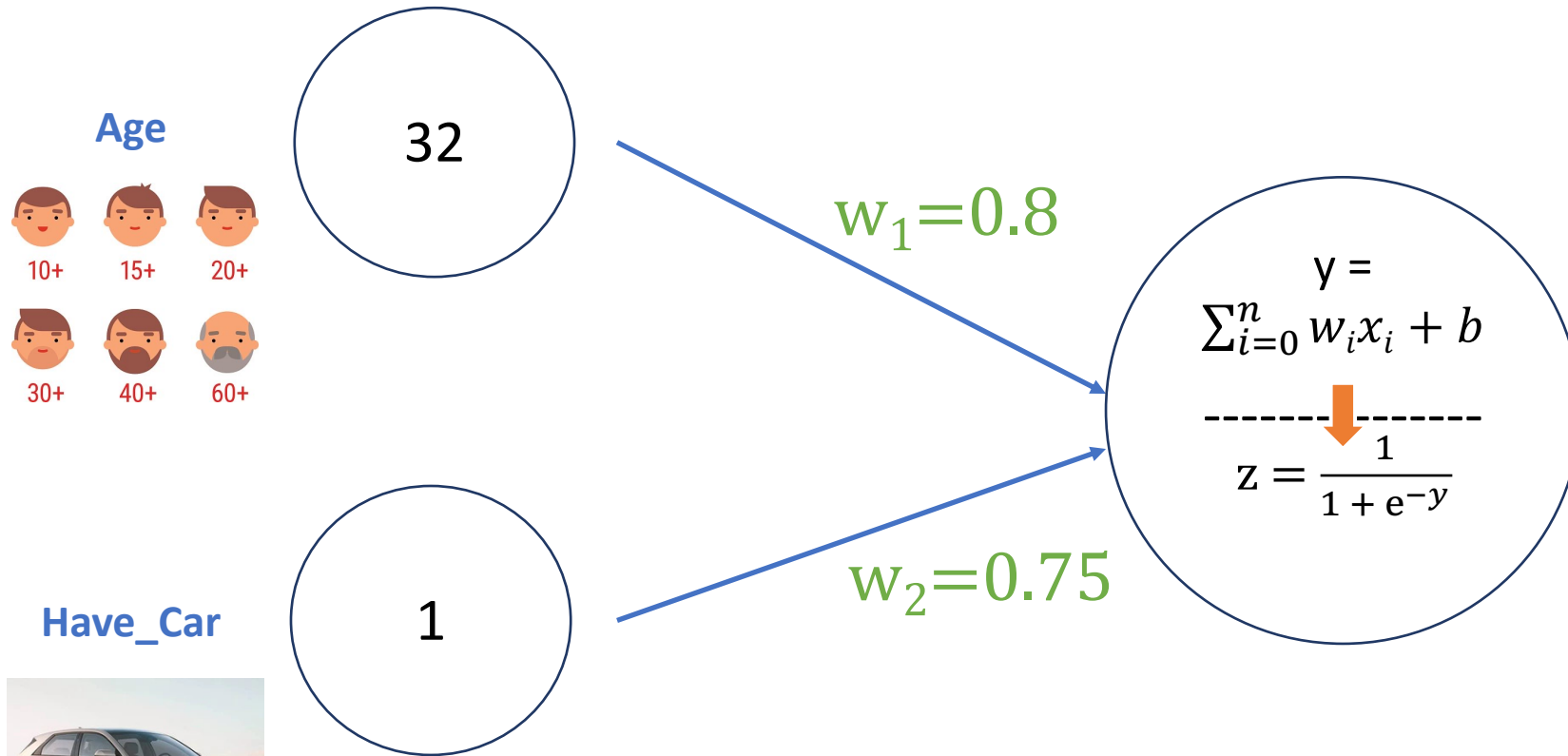
$$w_2 = 1$$

$$\text{bias} = 0$$

$$y = \sum_{i=0}^n w_i x_i + b$$

$$z = \frac{1}{1 + e^{-y}}$$

$$\text{Total Error} = \text{error}_1 + \text{error}_2 + \text{error}_3 + \dots$$



$$y = w_1 * X_1 + w_2 * X_2 + \text{bias}$$

$$y = 0.8 * X_1 + 0.75 * X_2 - 0.2$$

$$w_1 = w_1 - \text{learning rate} * \partial / \partial w_1$$

$$w_1 = 0.8 - 0.1 = 0.7$$

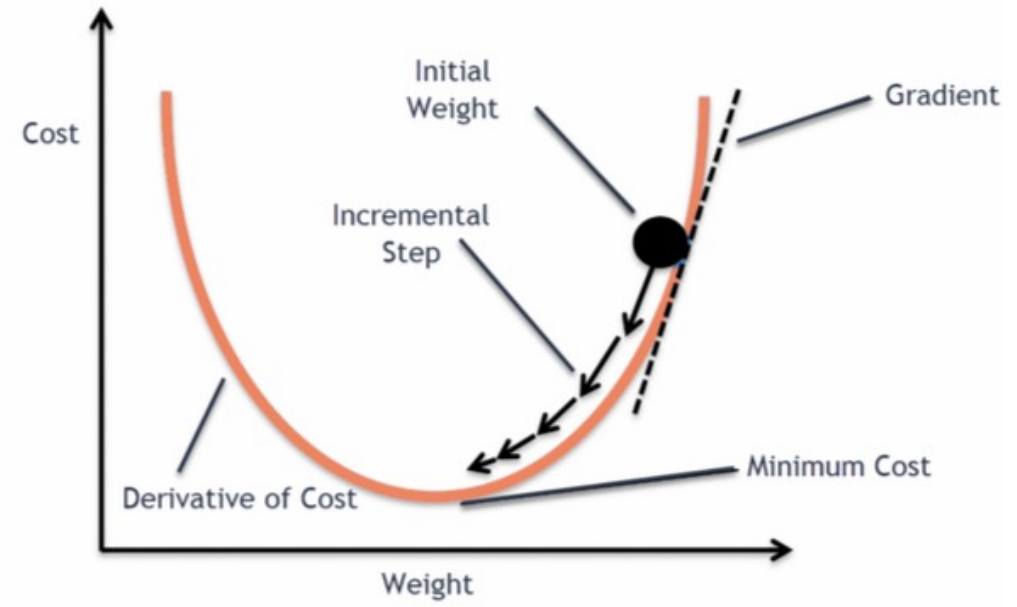
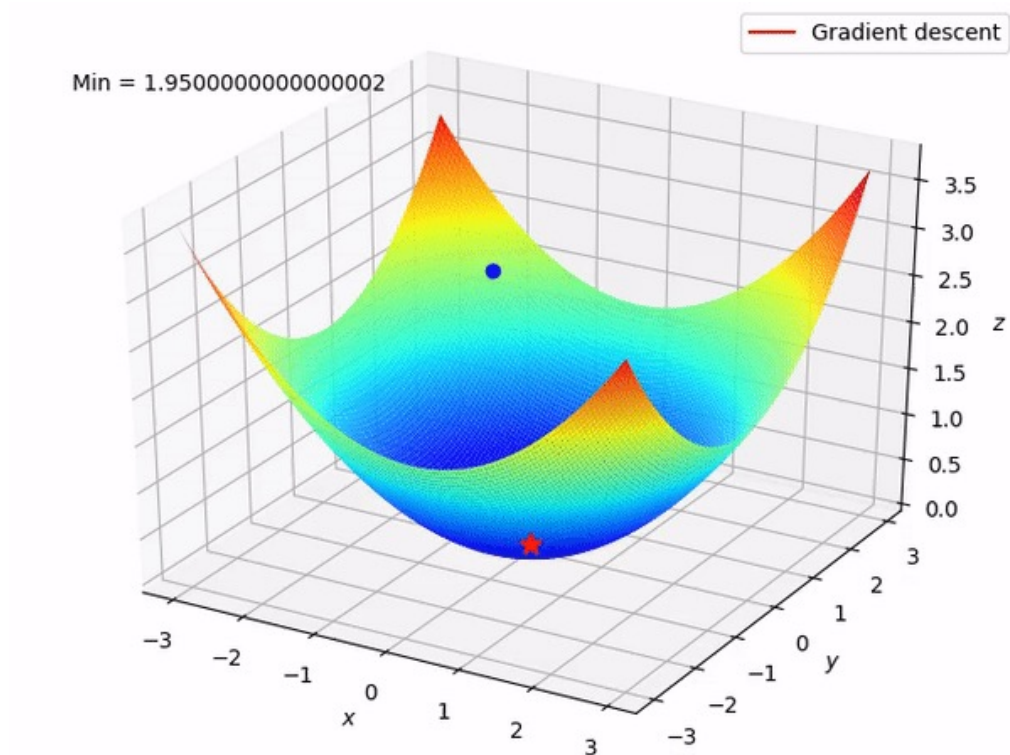
$$w_2 = w_2 - \text{learning rate} * \partial / \partial w_2$$

$$w_2 = 0.75 - 0.15 = 0.6$$

$$b = b - \text{learning rate} * \partial / \partial b$$

$$b = -0.2 - 0.1 = -0.3$$

Gradient Descent



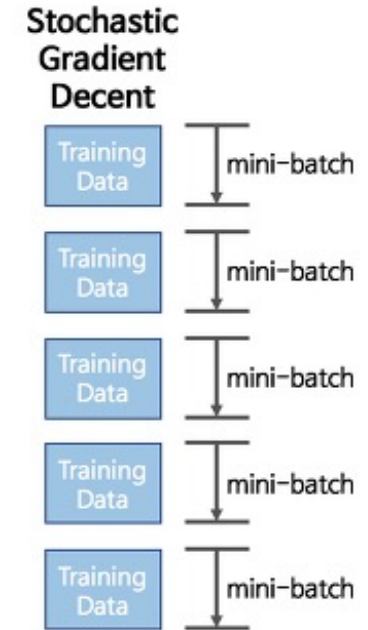
Gradient Descent



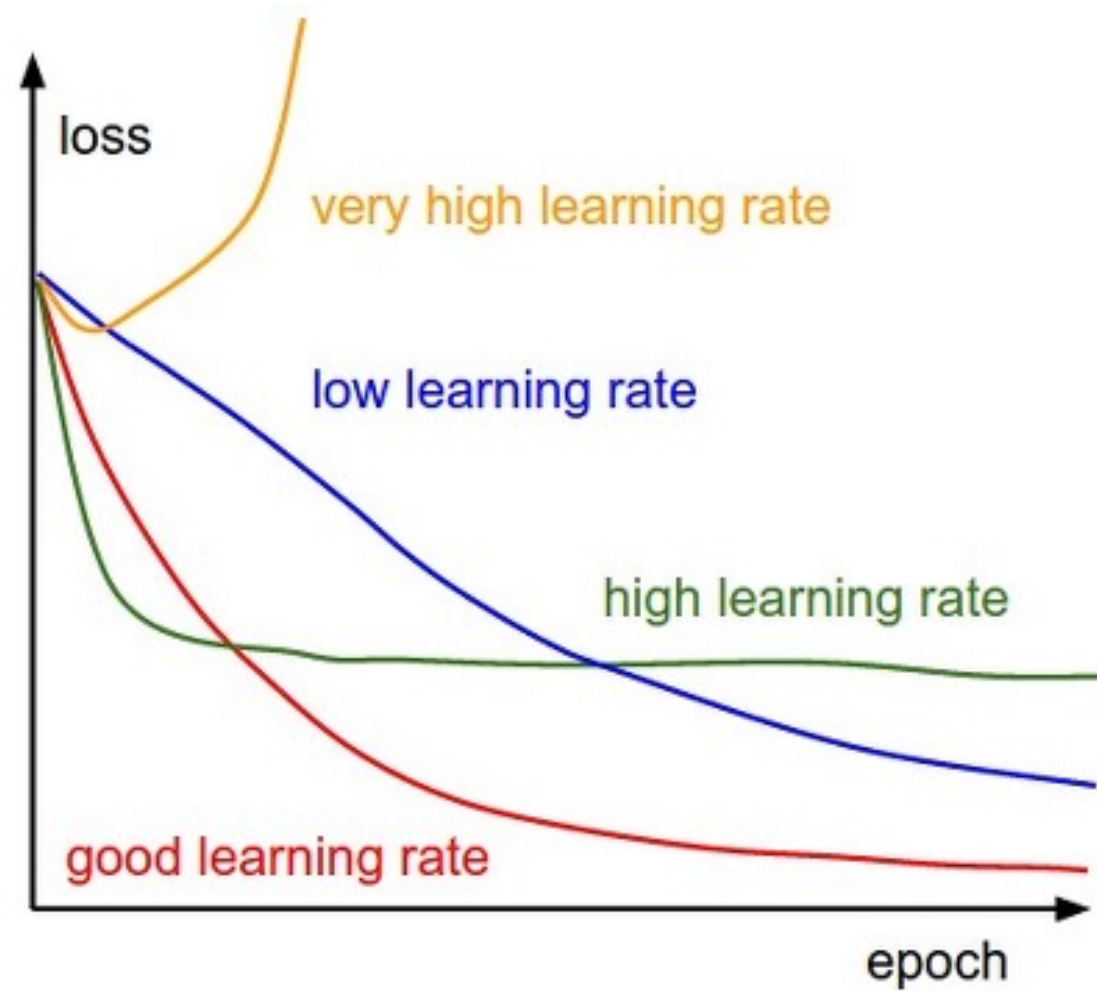
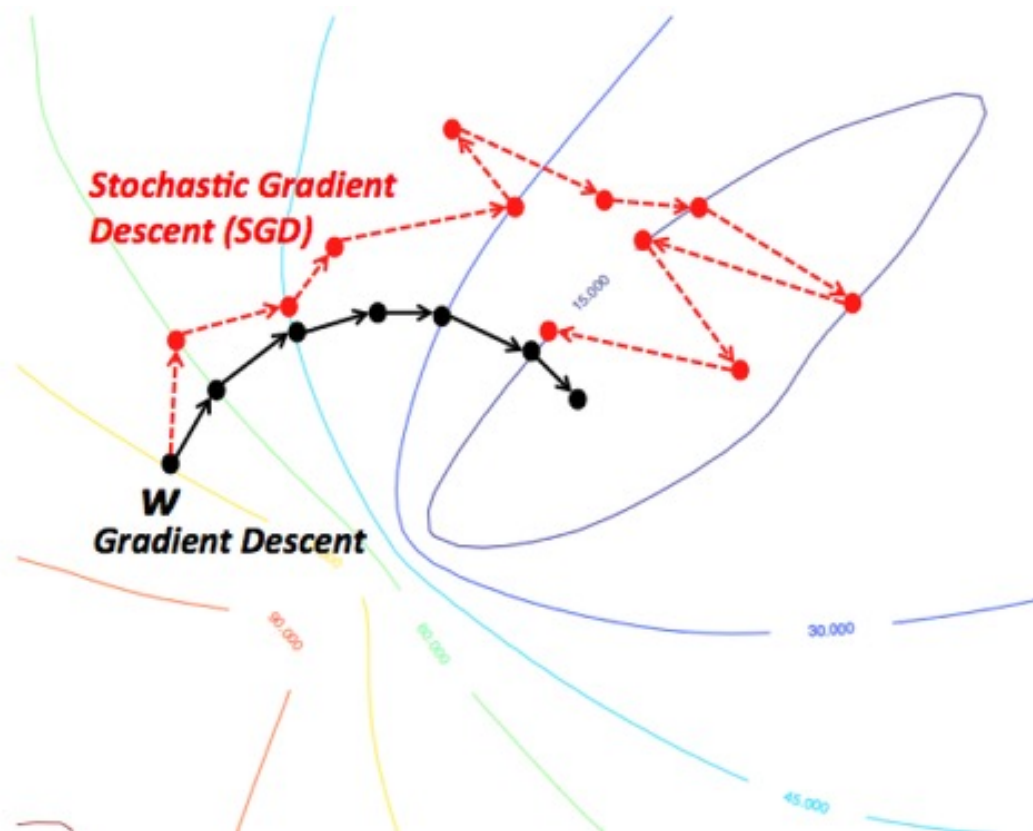
Batch Gradient Descent updates parameters for the **entire** training dataset.

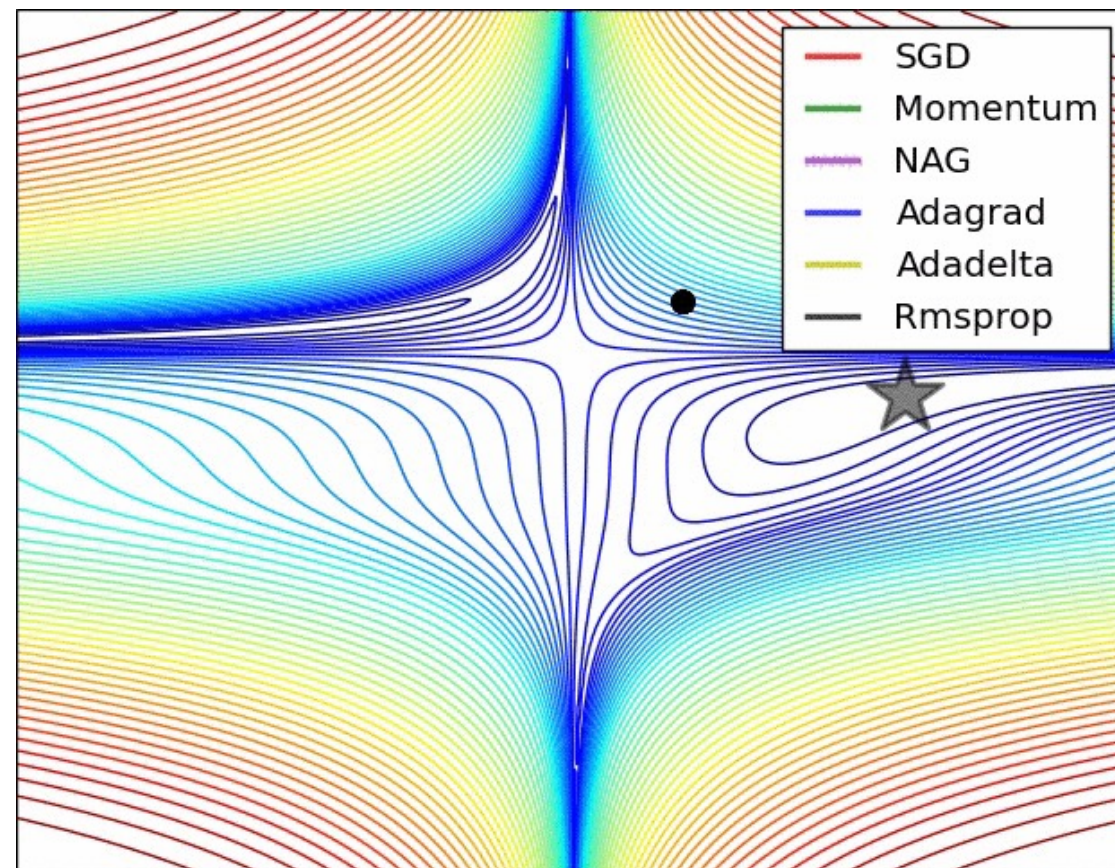
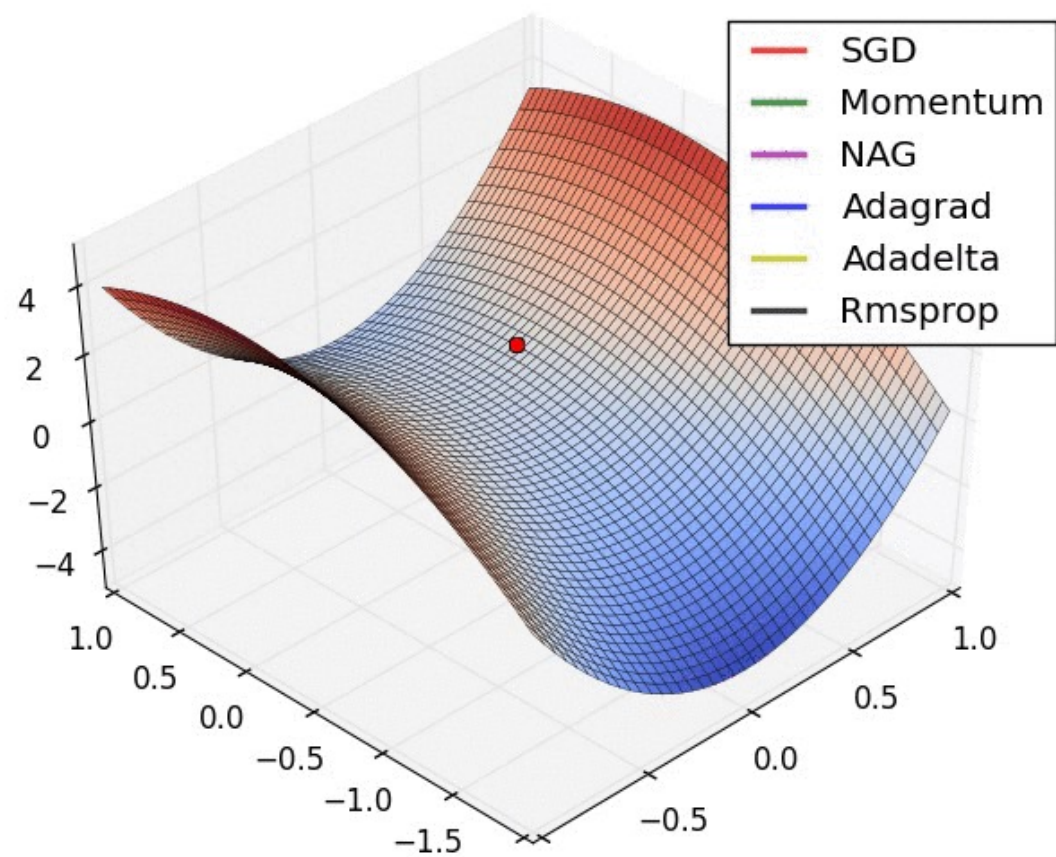


Stochastic Gradient Descent updates parameters for **each** training example and label.



Mini-batch Gradient Descent is a mixture of Batch Gradient Descent and Stochastic Gradient Descent.





Batch Gradient Descent (BGD)

```
for i in range(m):  
    gradient=evaluate_gradient(training_data)  
    weight=weight-learning_rate * gradient
```

Stochastic Gradient Descent (SGD)

```
for i in range(m):  
    np.random.shuffle(training_data)  
    for one_data in training_data:  
        gradient=evaluate_gradient(one_data)  
        weight=weight-learning_rate * gradient
```

Mini-batch Gradient Descent

```
for i in range(m):  
    np.random.shuffle(training_data)  
    for one_batch in get_mini_batches(training_data, one_batch_size=32):  
        gradient=evaluate_gradient(one_batch)  
        weight=weight-learning_rate * gradient
```

PyTorch: Optimizer

- `optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)`
- `optimizer = optim.Adam([var1, var2], lr=0.0001)`
(<https://pytorch.org/docs/stable/optim.html>)

TensorFlow-Keras: Optimizer

- `tf.keras.optimizers.SGD`
- `tf.keras.optimizers.Adam`
- <https://pytorch.org/docs/stable/optim.html#module-torch.optim>