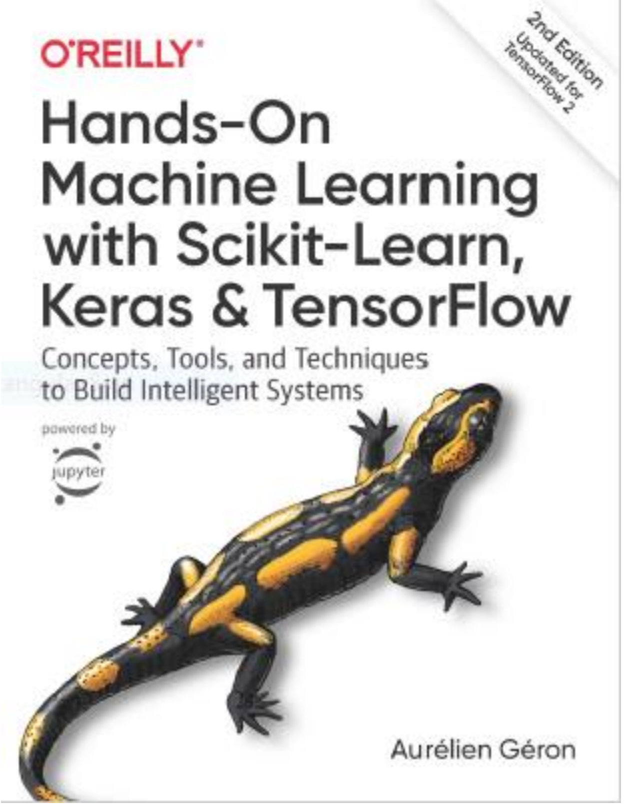
Training Models and Regression

Prof. Gheith Abandah

Reference



- Chapter 4: **Training Models**

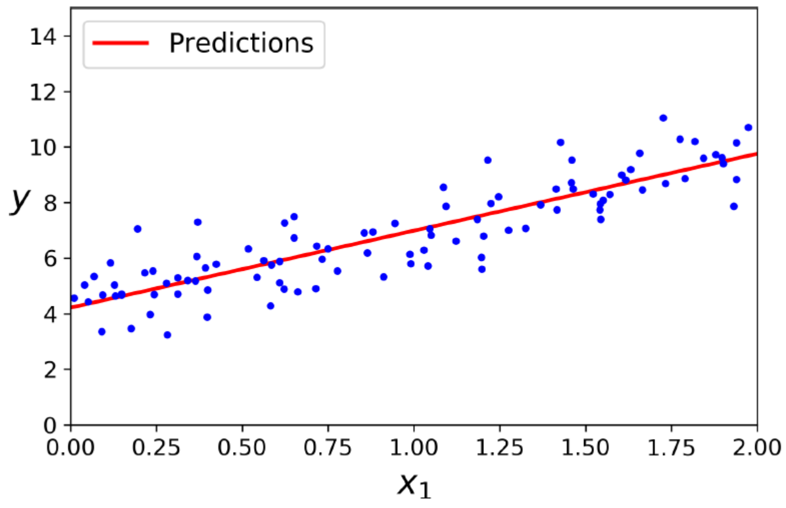
- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 2nd Edition, 2019
  - Material: https://github.com/ageron/handson-ml2

Outline

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
    1. Batch Gradient Descent
    2. Stochastic Gradient Descent
    3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

Linear Regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$



- $\hat{y}$ is the predicted value.
- n is the number of features.
- x_i is the ith feature value.
- θ_j is the jth model parameter (including the bias term θ_0 and the feature weights $\theta_1, \theta_2, \cdots, \theta_n$).

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

Analytical Solution

- The Root Mean Square Error (RMSE) is used as **cost function**.

$$\text{MSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

- Minimizing this cost gives the following solution (**normal function**):

$$\hat{\boldsymbol{\theta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$$

Complexity $O(mn^2)$

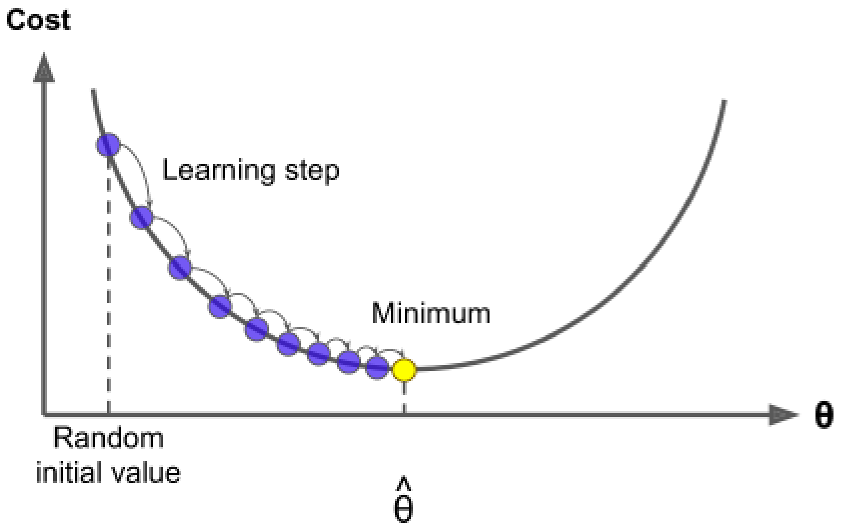
- $\hat{\boldsymbol{\theta}}$ is the value of $\boldsymbol{\theta}$ that minimizes the cost function.
- $\mathbf{y}$ is the vector of target values containing $y^{(1)}$ to $y^{(m)}$.

Outline

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
    1. Batch Gradient Descent
    2. Stochastic Gradient Descent
    3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

Gradient Descent

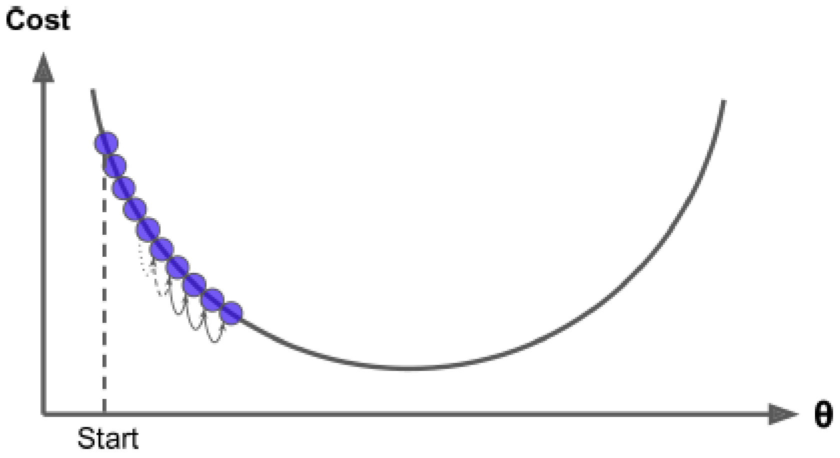
- **Generic optimization algorithm** capable of finding optimal solutions to a wide range of problems.
- **Tweaks parameters** iteratively in order **to minimize a cost function**.



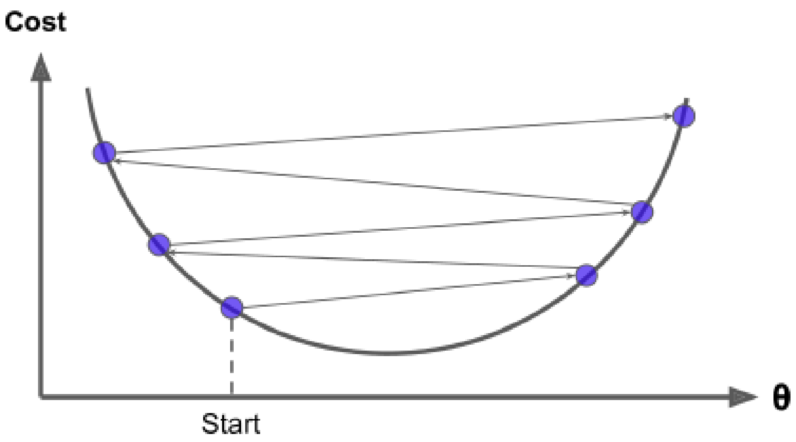
$$\theta^{(\text{next step})} = \theta - \eta \nabla_{\theta} \text{MSE}(\theta)$$

Learning Rate

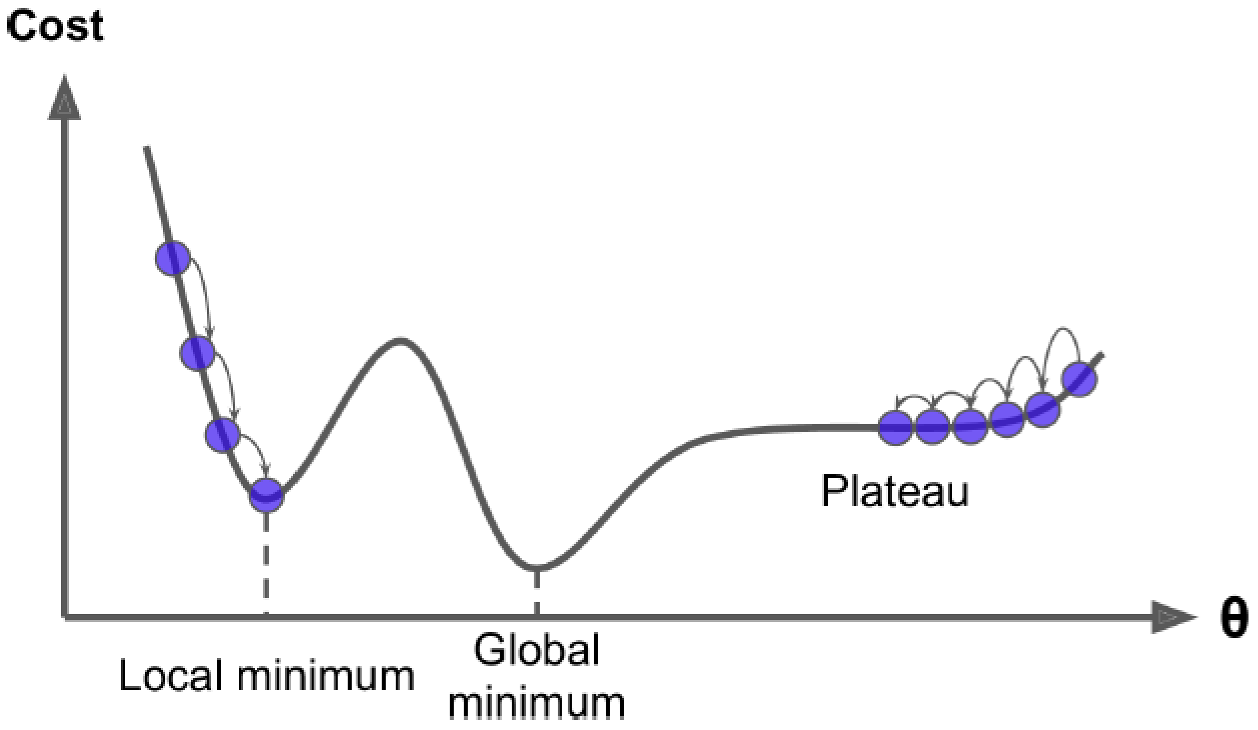
Too Small



Too Large

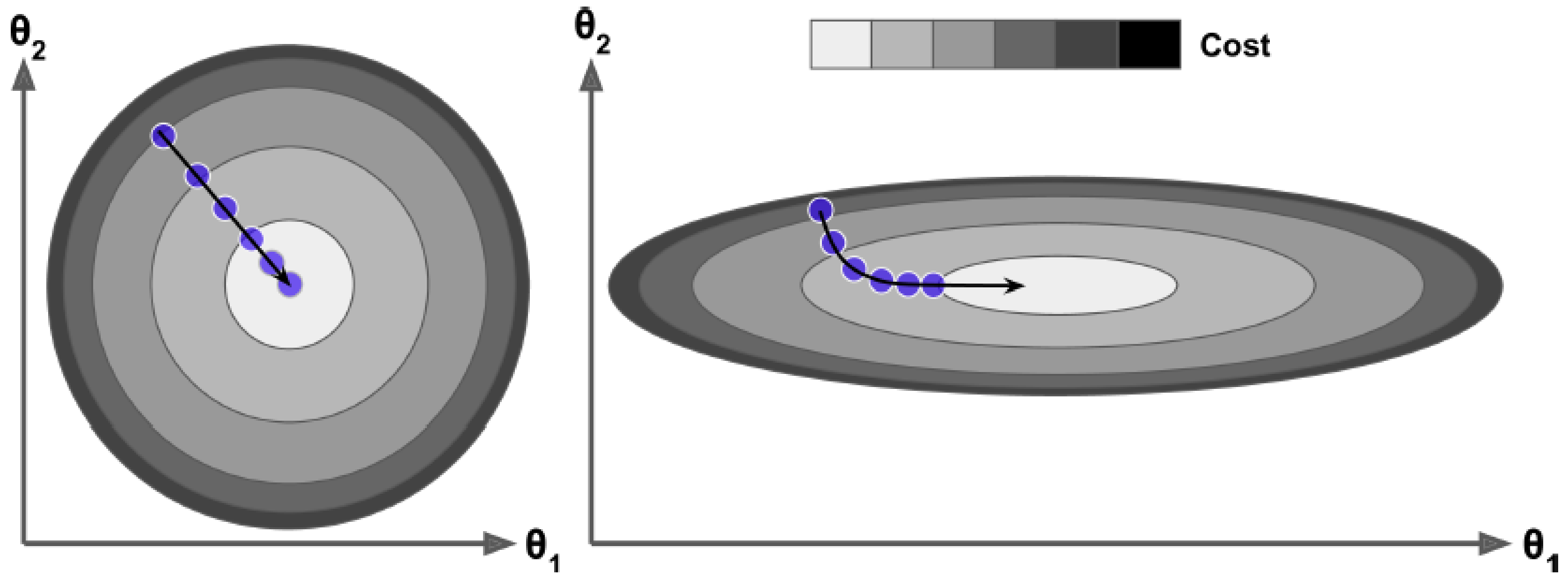


Gradient Descent Pitfalls



Feature Scaling

- Ensure that all features have a similar scale (e.g., using Scikit-Learn's `StandardScaler` class).
- Gradient Descent with and without feature scaling.



Outline

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
    1. Batch Gradient Descent
    2. Stochastic Gradient Descent
    3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

Batch Gradient Descent

- **Partial derivatives** of the cost function in θ_j

$$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\boldsymbol{\theta}) = \frac{2}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

- **Gradient vector** of the cost function

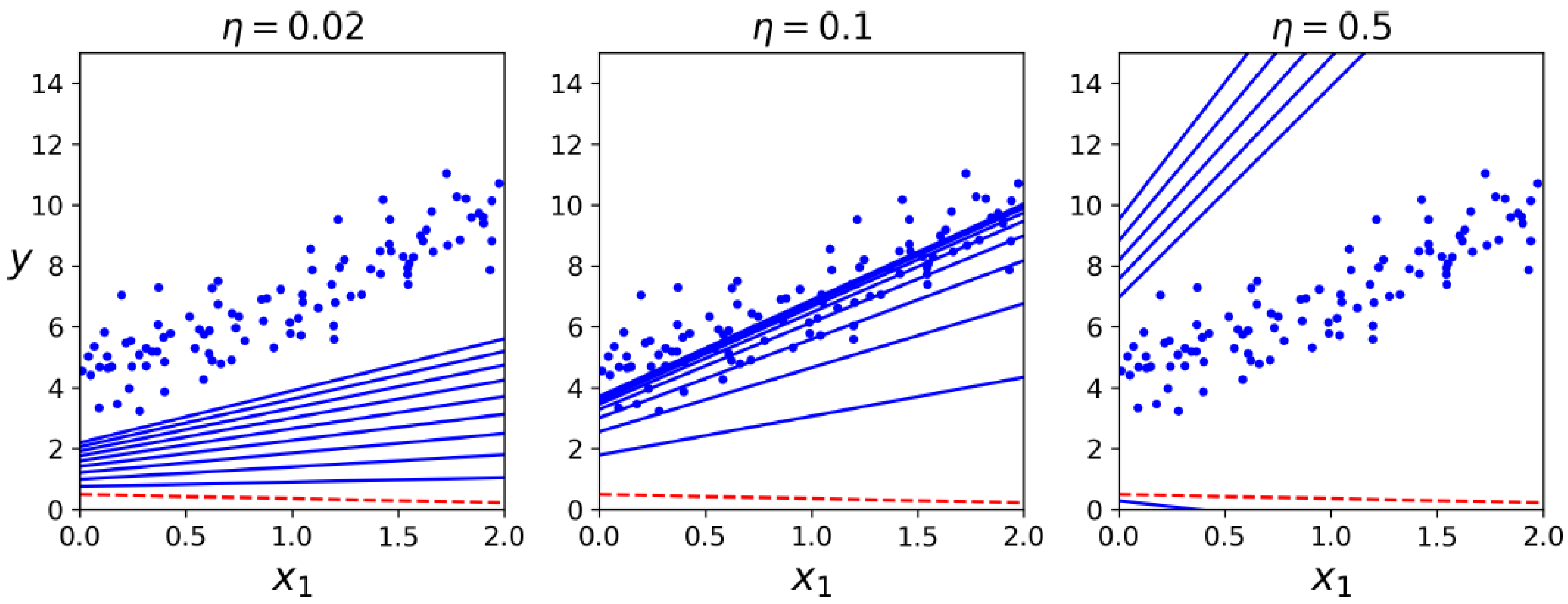
$$\nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \mathrm{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \mathrm{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \mathrm{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

The entire training Batch

$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta})$$

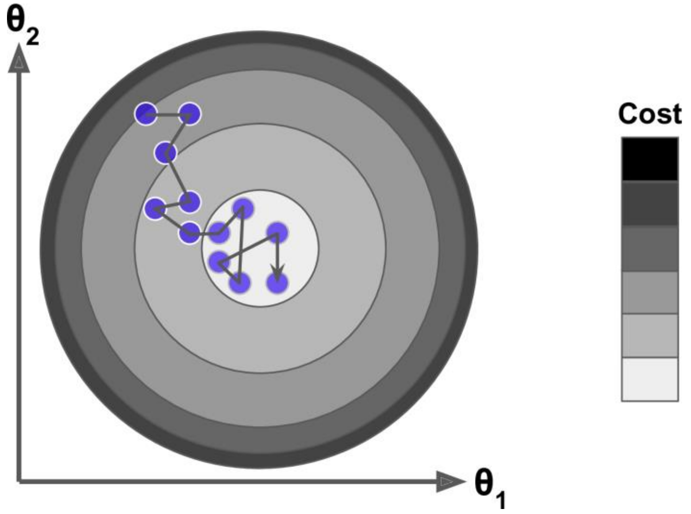
Batch Gradient Descent

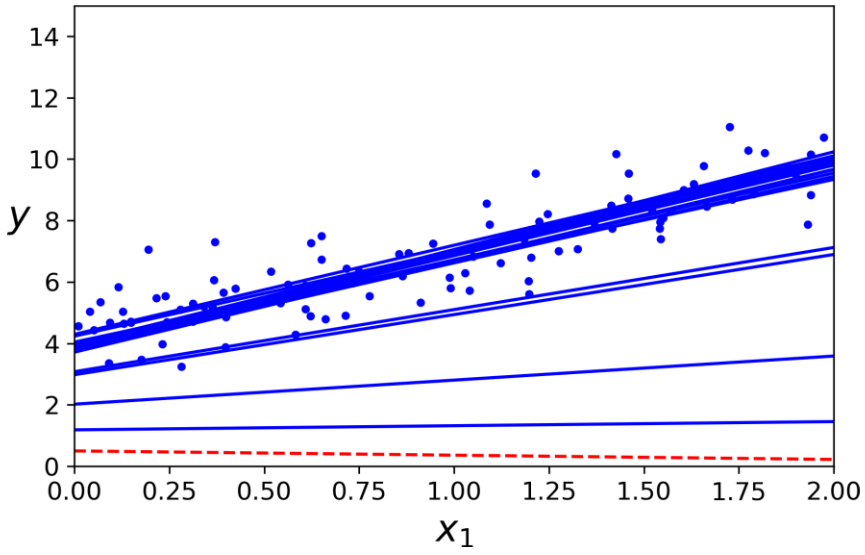
- **Gradient Descent step** $$\theta^{(\text{next step})} = \theta - \eta \nabla_{\theta} \text{MSE}(\theta)$$
- Gradient Descent with **various learning rates**



Stochastic Gradient Descent

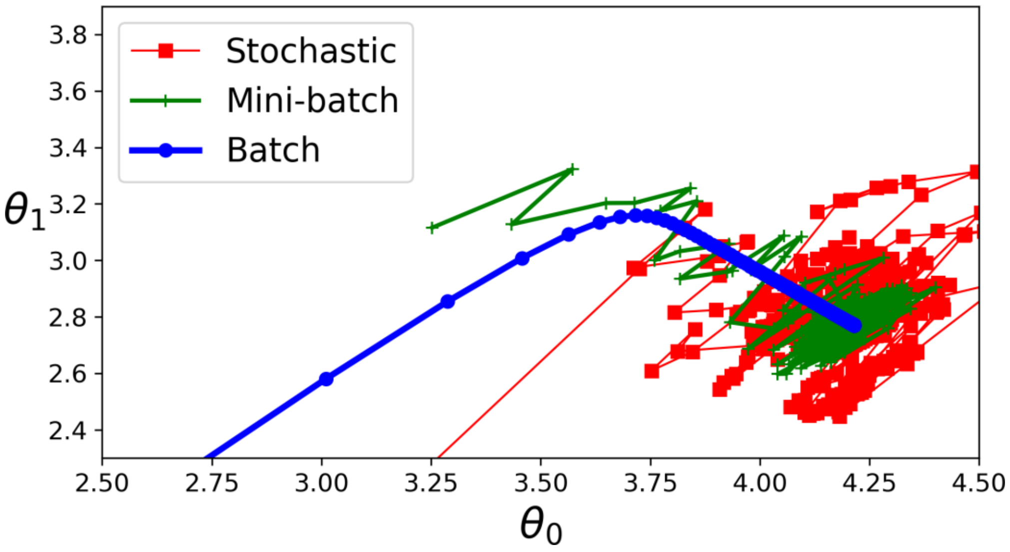
- SGD **picks a random instance** in the training set at every step and computes the gradients.
- SGD is **faster** when the training set is large.
- Is **bouncy**
- Eventually gives **good solution**
- Can **escape local minima**





Mini-batch Gradient Descent

- Computes the gradients on small random sets of instances called **mini batches**.
- Benefits from **hardware accelerators** (e.g., GPU).
- **Less bouncy**, **better solution**, **escapes some local minima**

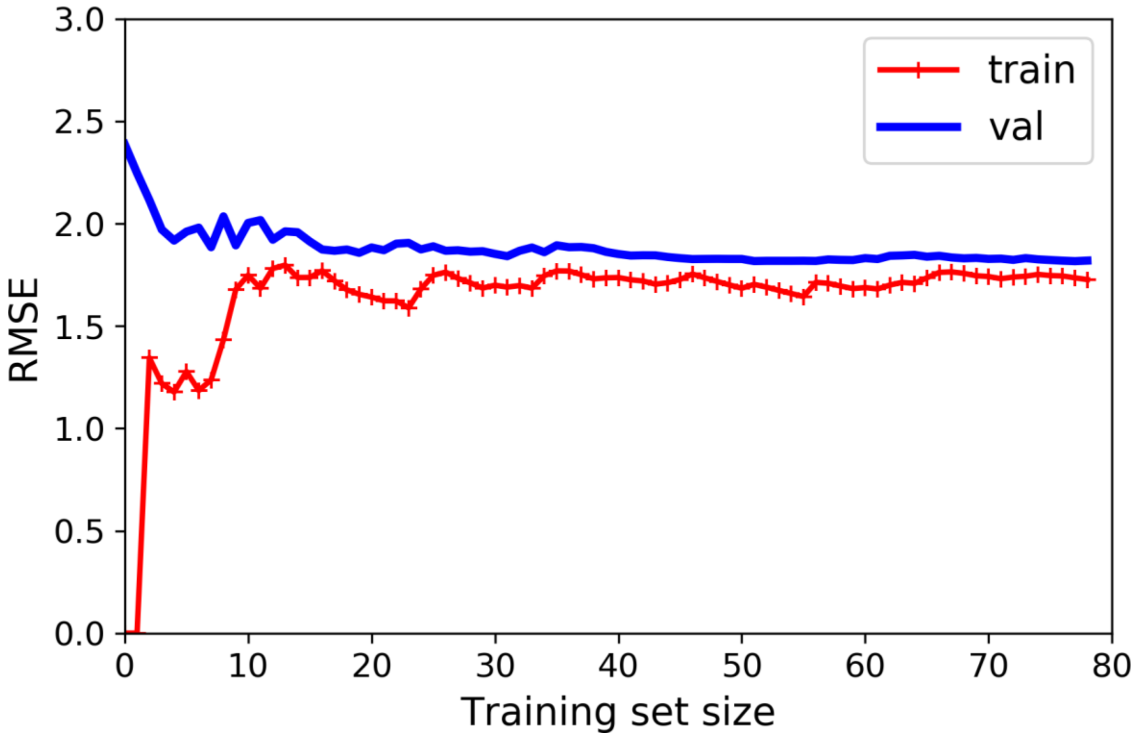


Outline

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
    1. Batch Gradient Descent
    2. Stochastic Gradient Descent
    3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

Learning Curves

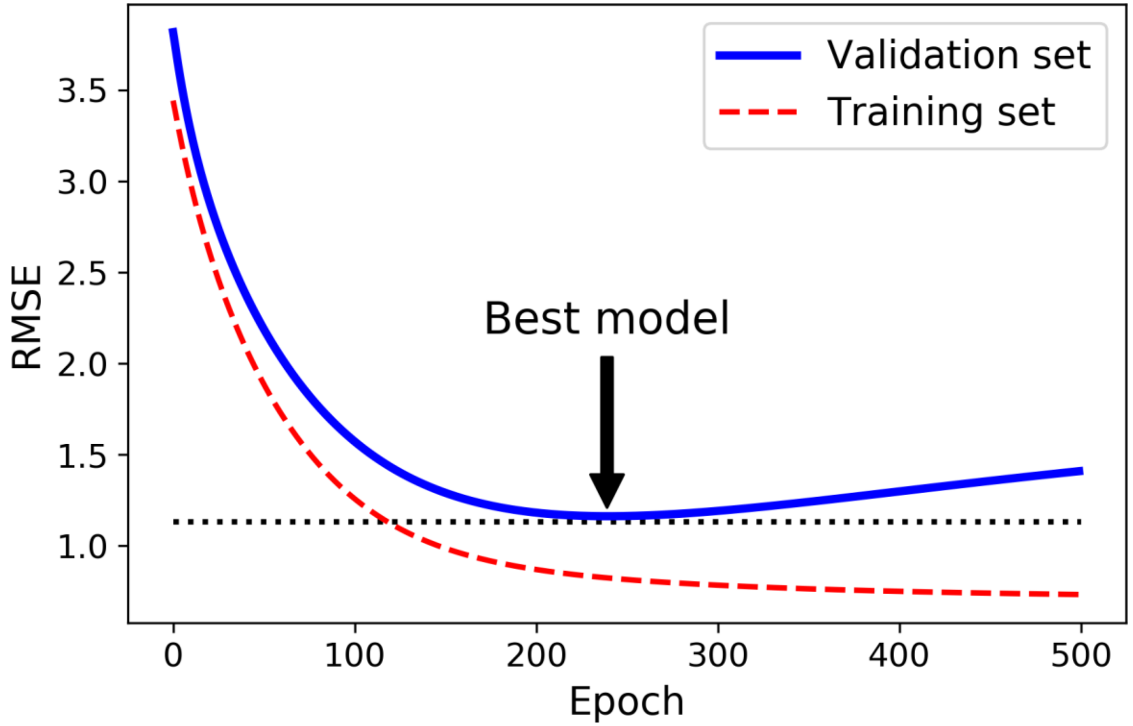
- The **accuracy** on the **validation set** generally **increases** as the **training set** size increases.
- **Overfitting decreases** with larger training set.



Outline

Early Stopping

- **Stop** training when the **validation error reaches a minimum**.
- Need to **save the best model**.



Outline

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
    1. Batch Gradient Descent
    2. Stochastic Gradient Descent
    3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises

Exercises

1. What Linear Regression training algorithm can you use if you have a training set with millions of features?
2. Suppose the features in your training set have very different scales. What algorithms might suffer from this, and how? What can you do about it?
3. Do all Gradient Descent algorithms lead to the same model provided you let them run long enough?

Summary

1. Linear Regression
2. Gradient Descent
3. Gradient Descent Variants
    1. Batch Gradient Descent
    2. Stochastic Gradient Descent
    3. Mini-batch Gradient Descent
4. Learning Curves
5. Early Stopping
6. Exercises