Training Models 1: Gradient Descent

Glenn Bruns
CSUMB

Learning outcomes

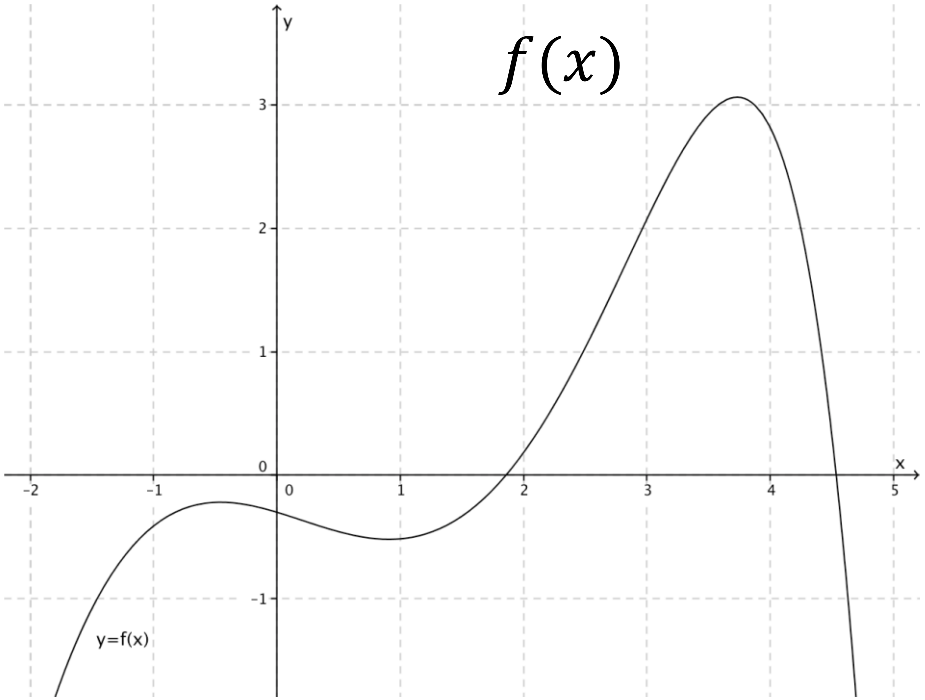
After this lecture you should be able to:

1. List and explain various methods for finding the maximum of a numeric function
2. Compare these methods
3. Explain how these methods can be used with linear regression

Find the maximum value of a function

Problem:

- There is a function f
- We want to find the value x such that $f(x)$ is the maximum value of f
- We may or may not know the definition of f



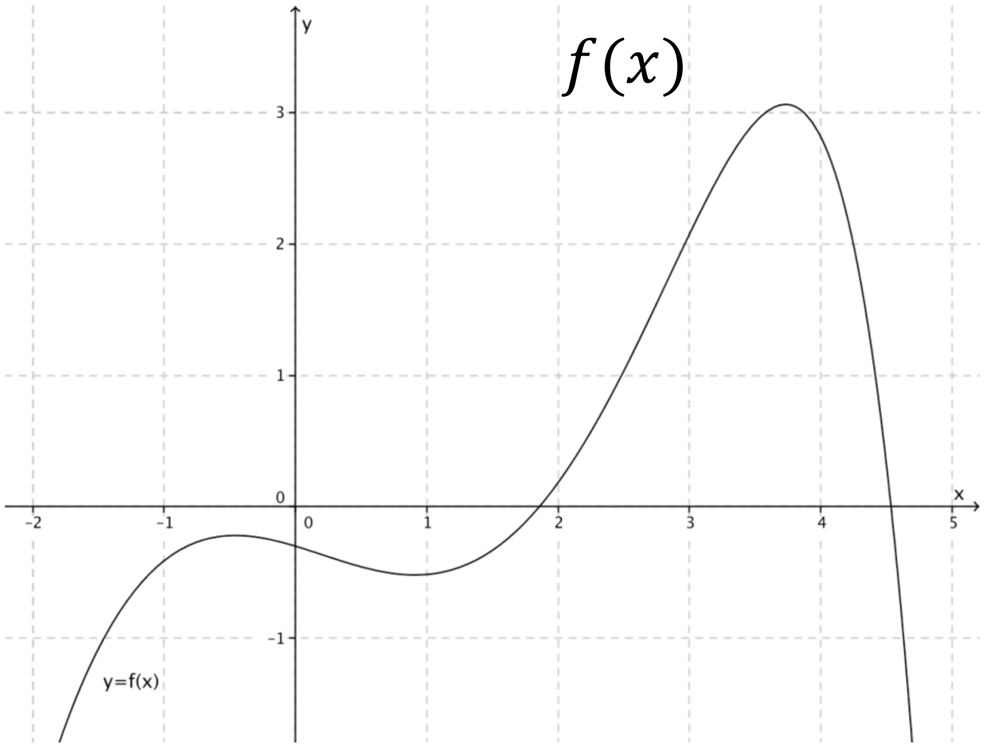
How to find this x?

Idea 1: grid search

concept: sample f at regular intervals

What is the result from the figure?

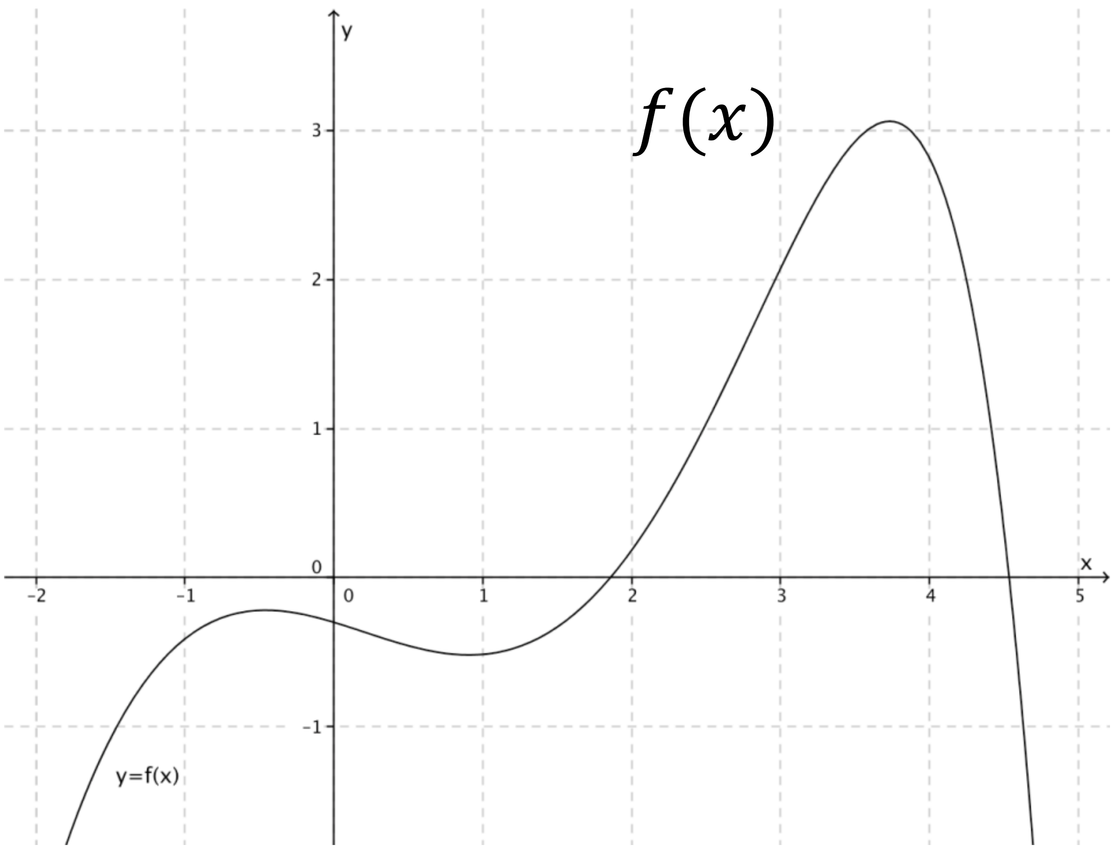
What are the pros and cons?



Idea 2: use derivatives

concept: find values x such that $f'(x) = 0$

(first derivative test)

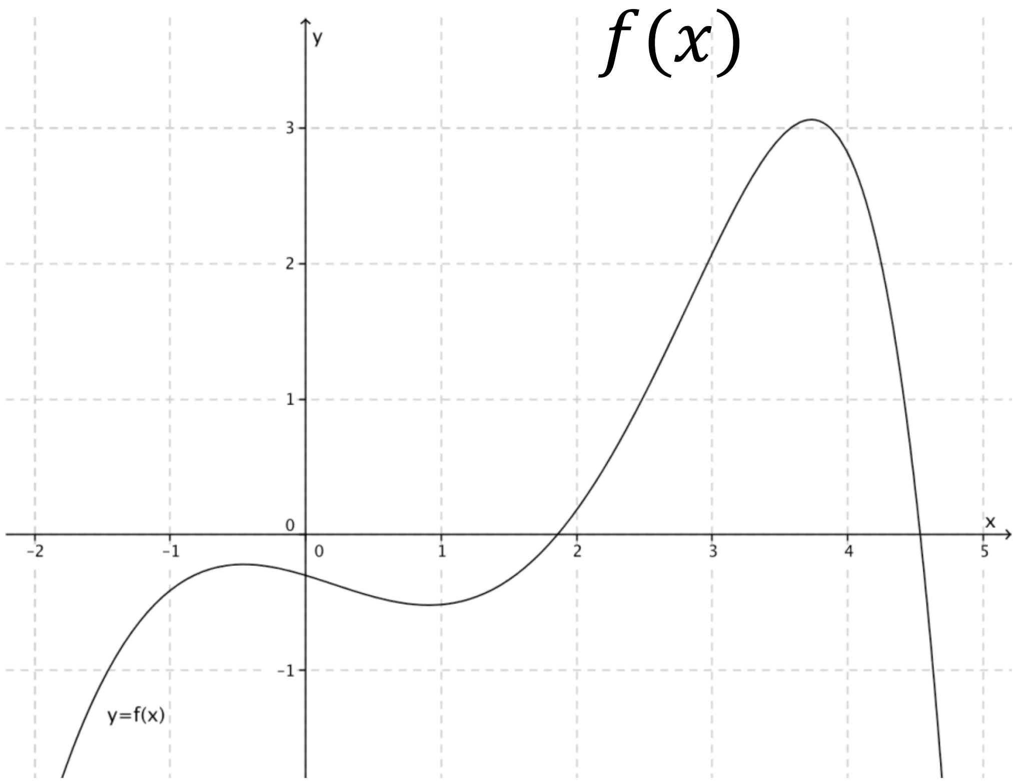


Idea 3: gradient descent, analytic version

concept:

- start at a random x value
- change x by "moving uphill": modify x based on the slope of f at x, i.e. $f'(x)$

What is the formula for updating x?



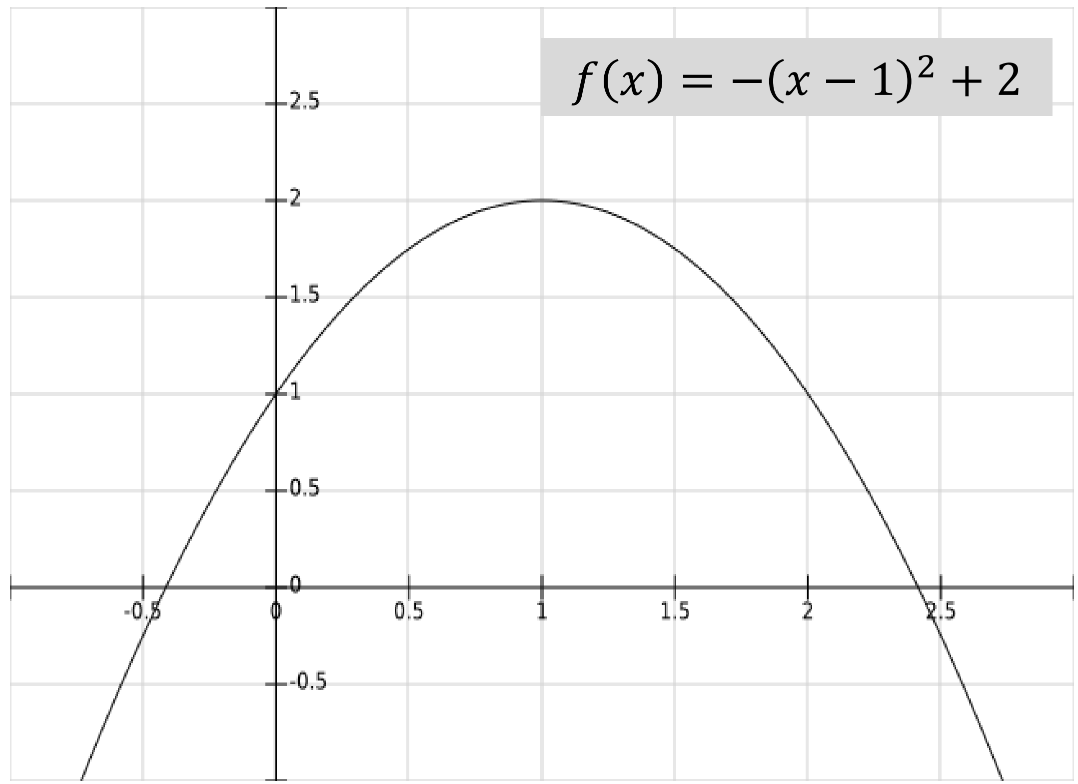
Gradient descent example

Derivative of x:

$$f'(x) = -2(x - 1)$$

To update x:

$$x := x + \eta f'(x)$$

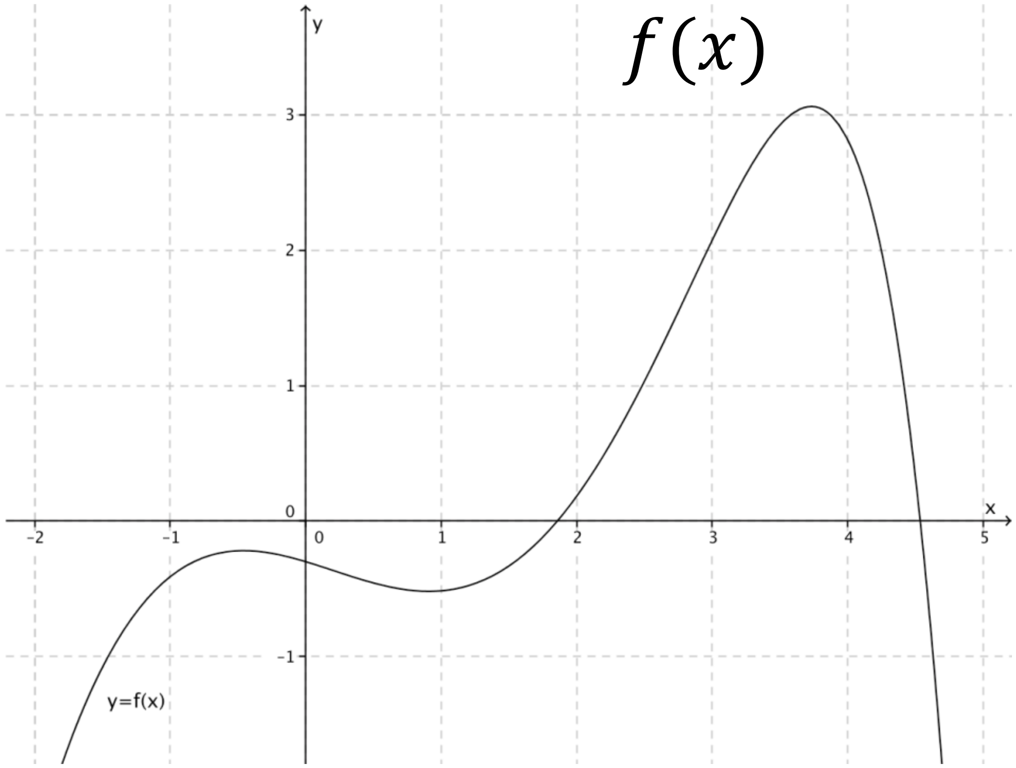


Idea 4: gradient descent, numeric version

concept:

- similar to version in which we know the derivative of f
- here we estimate the derivative at a point x numerically

$f(x)$



Linear regression

Prediction in linear regression:

$$\hat{y} = \theta^T \cdot \mathbf{x}$$

θ is the model's parameter vector

$\mathbf{x}$ is the feature vector ($\mathbf{x}_0$ is always 1)

$\hat{y}$ is the estimated (predicted) value of y

MSE cost function for linear regression:

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

Reminder: Geron writes $\mathbf{x}^{(i)}$ for the ith row of matrix X.

Given training data $\mathbf{X}$, we want the value of θ that *minimizes* $MSE(\mathbf{X}, \theta)$

Closed-form solution

You can calculate the model parameters directly using the Normal Equation for linear regression:

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

$\hat{\theta}$ is the estimated model parameter vector

$\mathbf{y}$ are the target values (labels) for training data $\mathbf{X}$

NumPy code to compute $\hat{\theta}$:

```
X = 2 * np.random.rand(100,1)
y = 4 + 3*X + np.random.randn(100,1)
X_b = np.c_[np.ones((100,1)), X] # X augmented with x0 = 1

theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

material on this slide closely follows Géron

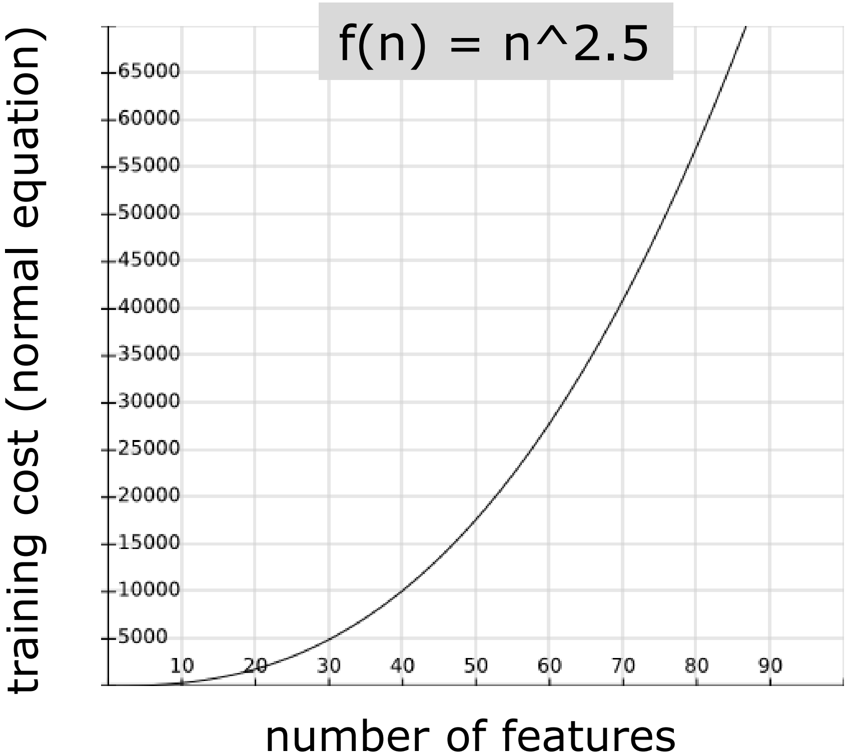
Why not always used closed-form solution?

- closed-form solution looks good
- Isn't it always better than using something like gradient descent?

Cost of computing the inverse of $\mathbf{X}^T \cdot \mathbf{X}$ is high. About $O(n^{2.4})$ to $O(n^3)$

If number of features doubles, time to compute goes up by about 8.

Good news: cost is linear in number of training examples.



Linear regression for Boston housing

Normal equation gives model parameters of (-30.0, 8.3).

A contour plot of RSS as a function of model parameters β_0, β_1

Dark red is high value, dark blue is low value.

Gradient descent in higher dimensions

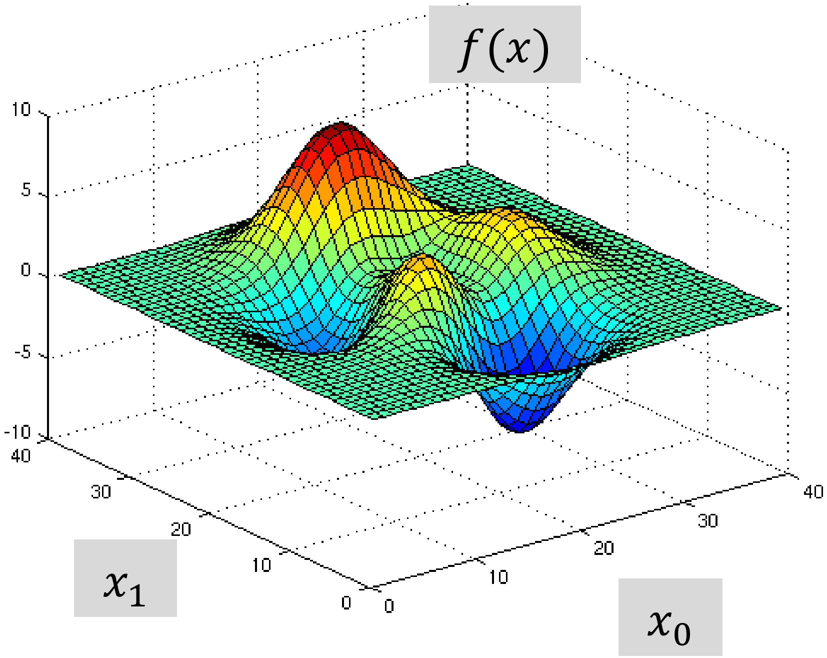
In functions of more than one dimension, we need to look at the slope in each dimension.

The multi-dimension generalization of the derivative is the gradient.

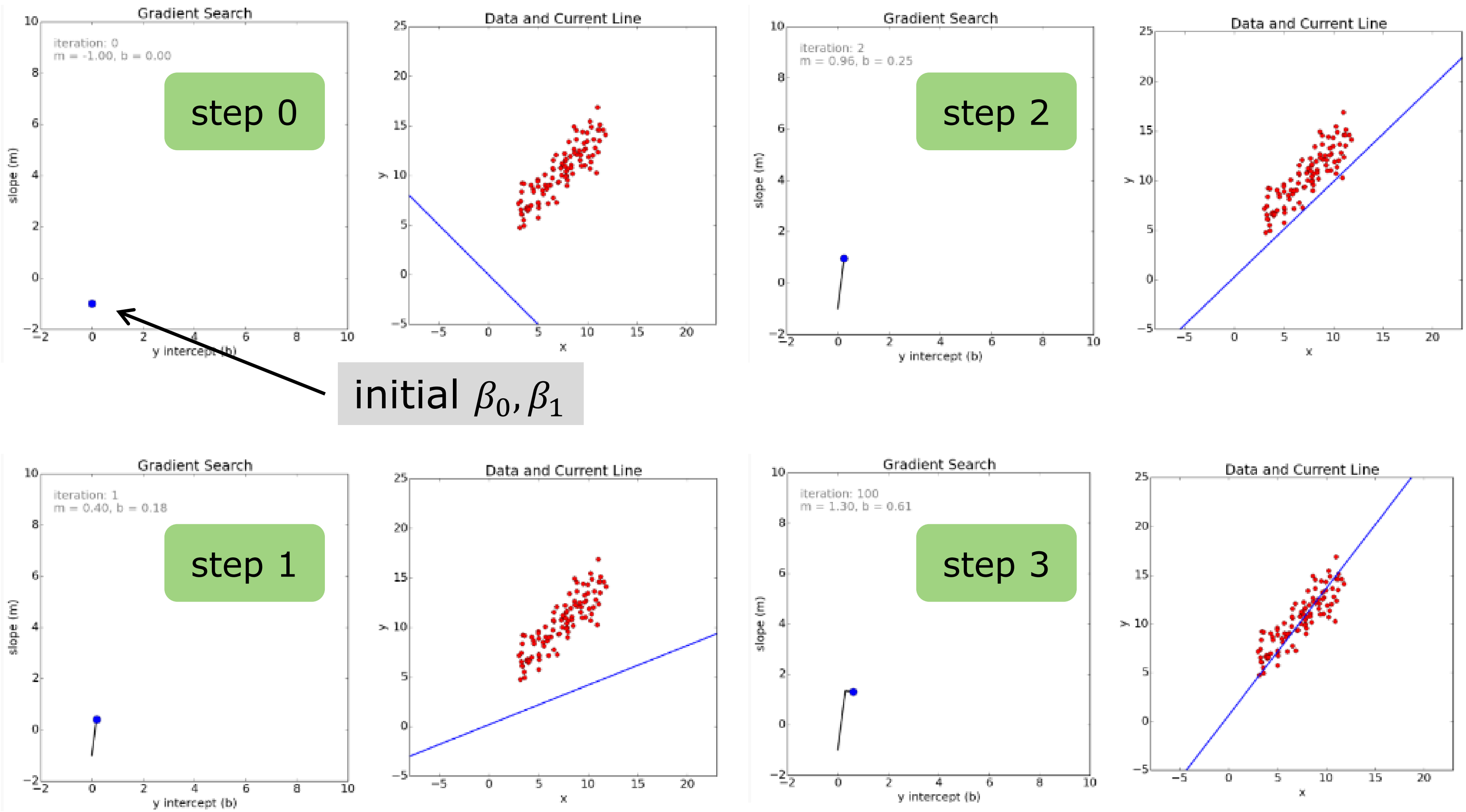
The gradient of f is written ∇f.

Multi-dimensional gradient descent:

$$x \coloneqq \theta - \eta \nabla f(x)$$

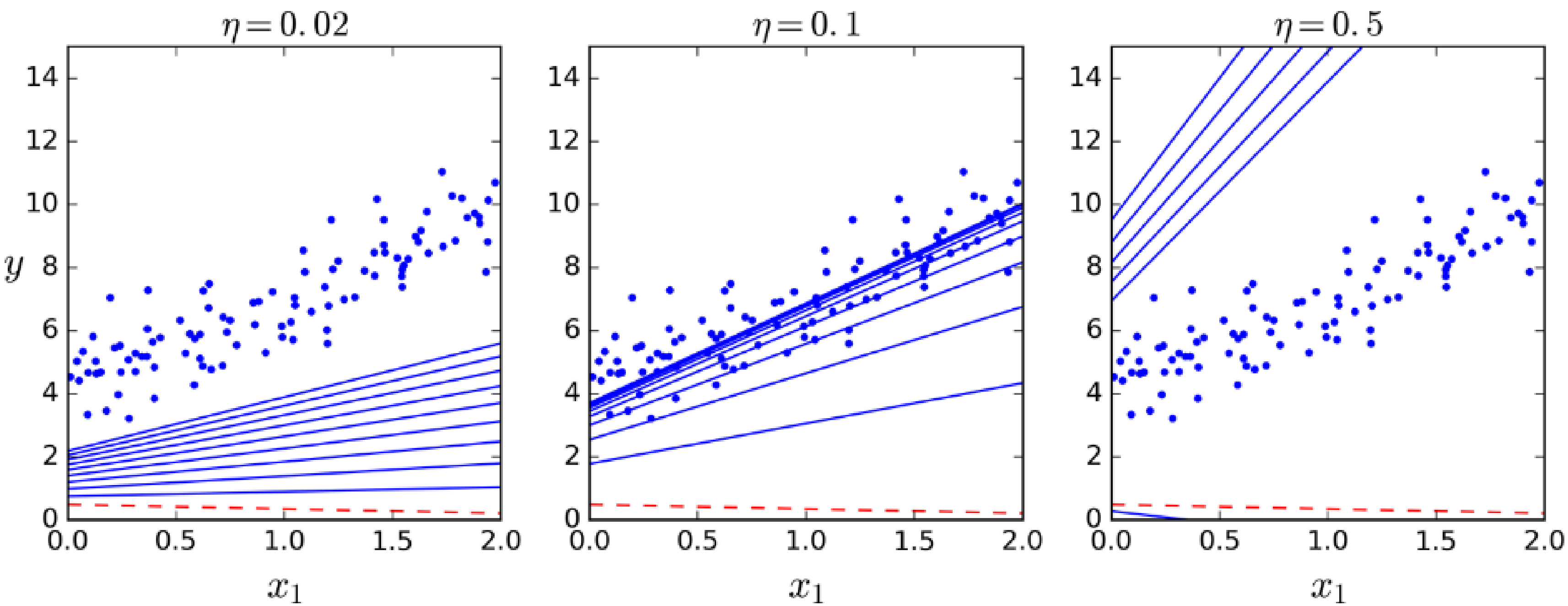


Gradient descent for linear regression



Impact of learning rate

figures show gradient descent with various values of learning rate



learning rate too slow

learning rate too fast

Summary

- How to finding the max (or min) value of a function?
  - grid search
  - use derivatives
  - gradient descent with analytical derivatives
  - gradient descent with numeric derivatives
- Linear regression as an optimization problem
  - gradient descent for functions of more than one variable