Artificial Intelligence

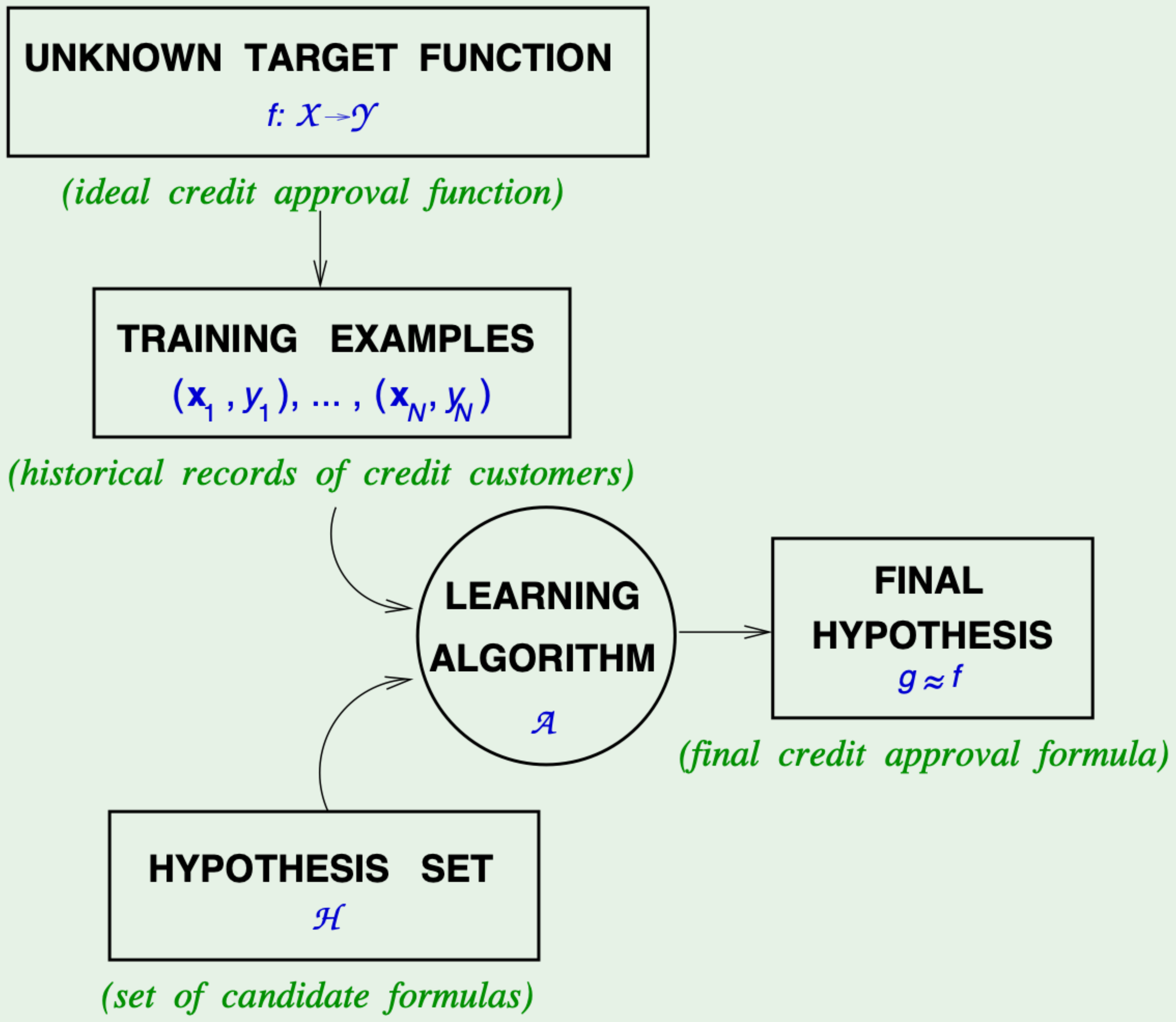
CSC 665

tyler dae devlin

Machine Learning III

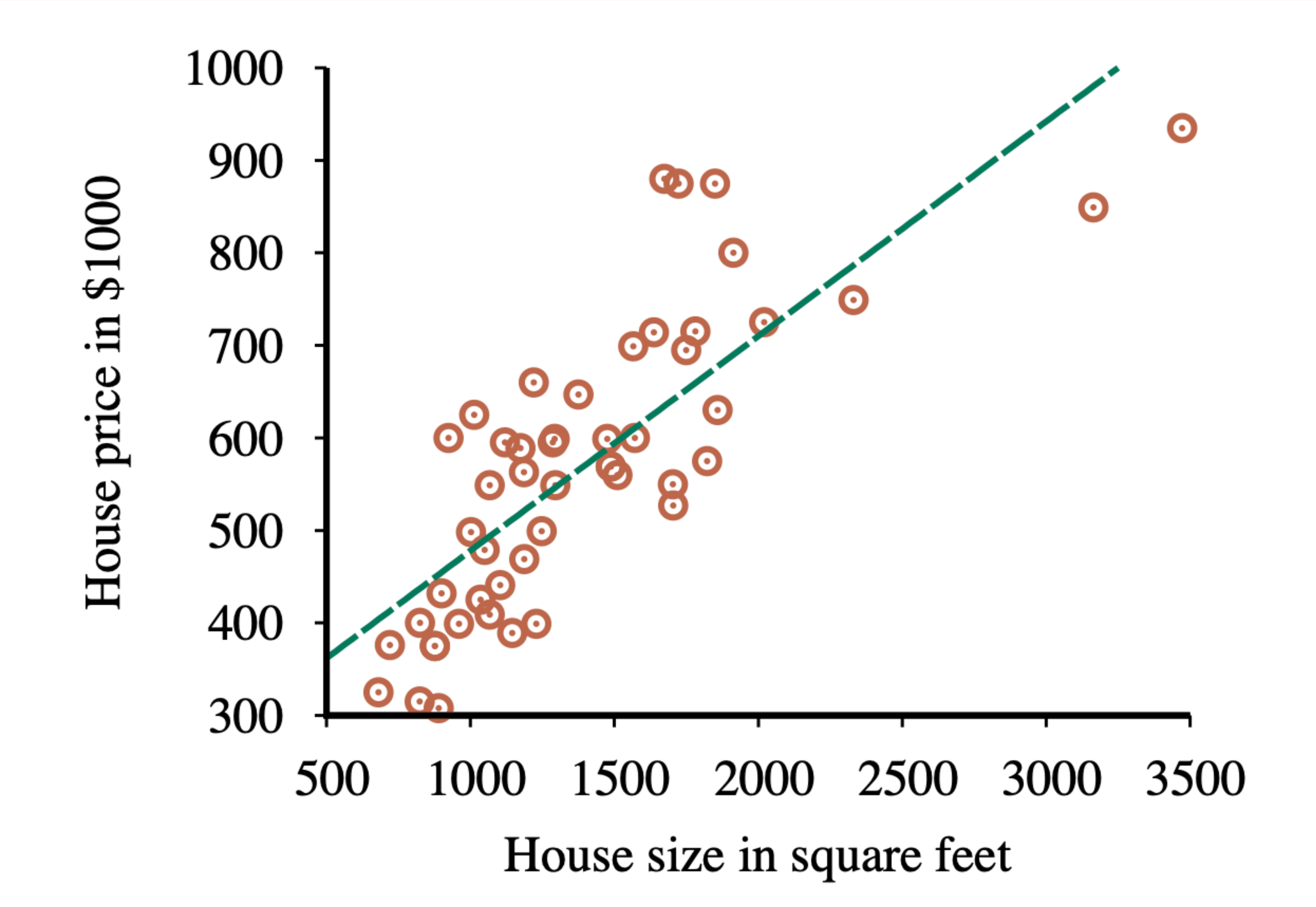
11.2.2023

- **Search:** make decisions by looking ahead
- **Logic:** deduce new facts from existing facts
- **Constraints:** find a way to satisfy a given specification
- **Probability:** reason quantitatively about uncertainty
- **Learning:** make future predictions from past observations


UNKNOWN TARGET FUNCTION
$f: \mathcal{X} \to \mathcal{Y}$
(ideal credit approval function)
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
(historical records of credit customers)
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
(final credit approval formula)
HYPOTHESIS SET
$\mathcal{H}$
(set of candidate formulas)

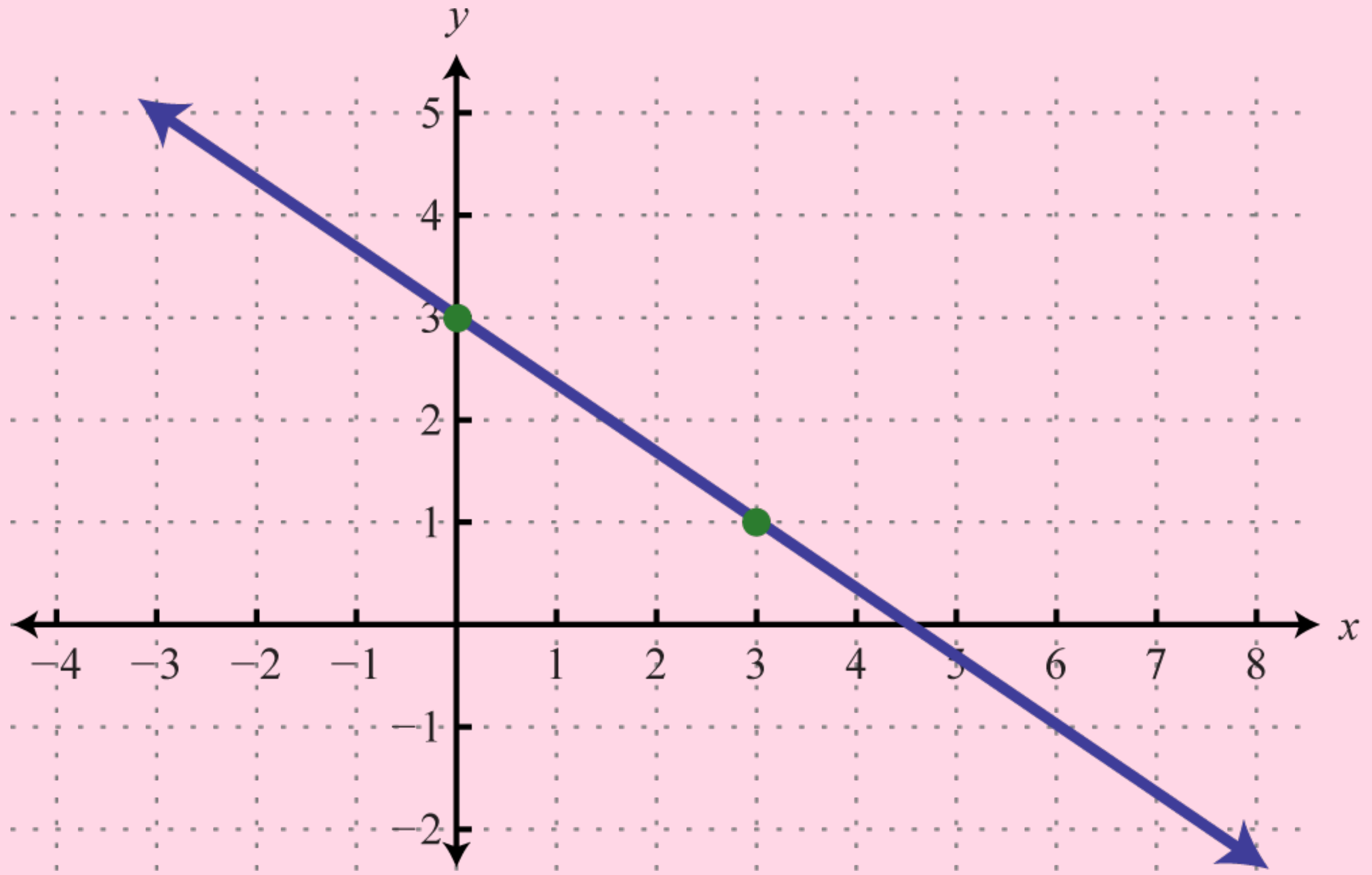
Modeling

Example: predicting housing prices



Linear models

- **Hypothesis class:** the set of all linear functions $h(x) = w_1 x + w_0$
- **Cost:** squared error
$$C(h) = \sum_{i=1}^{n} (y_i - h(x_i))^2$$
- **Optimizer:** analytical solutions for w_0, w_1 via calculus



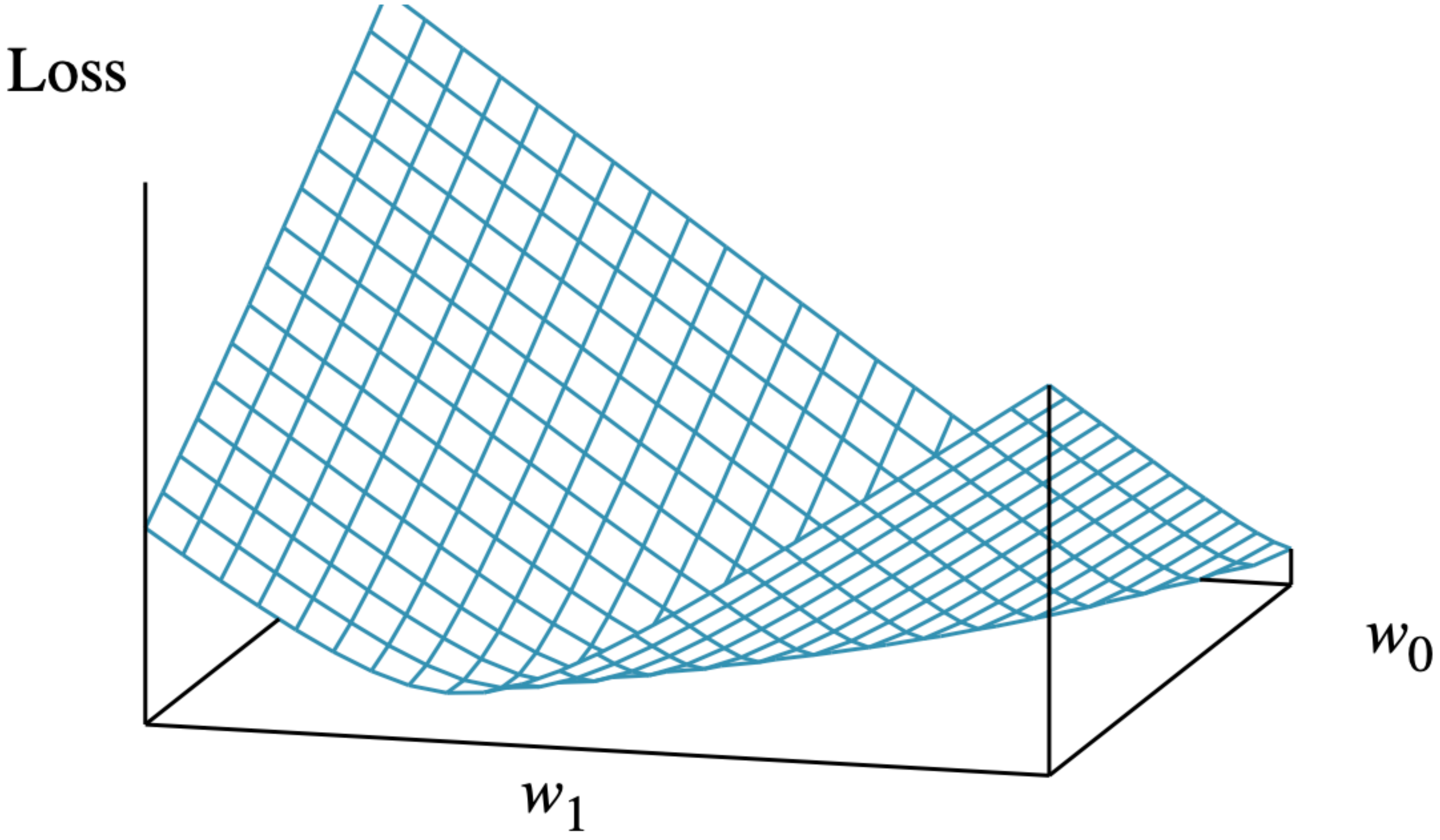
Optimizer: calculus

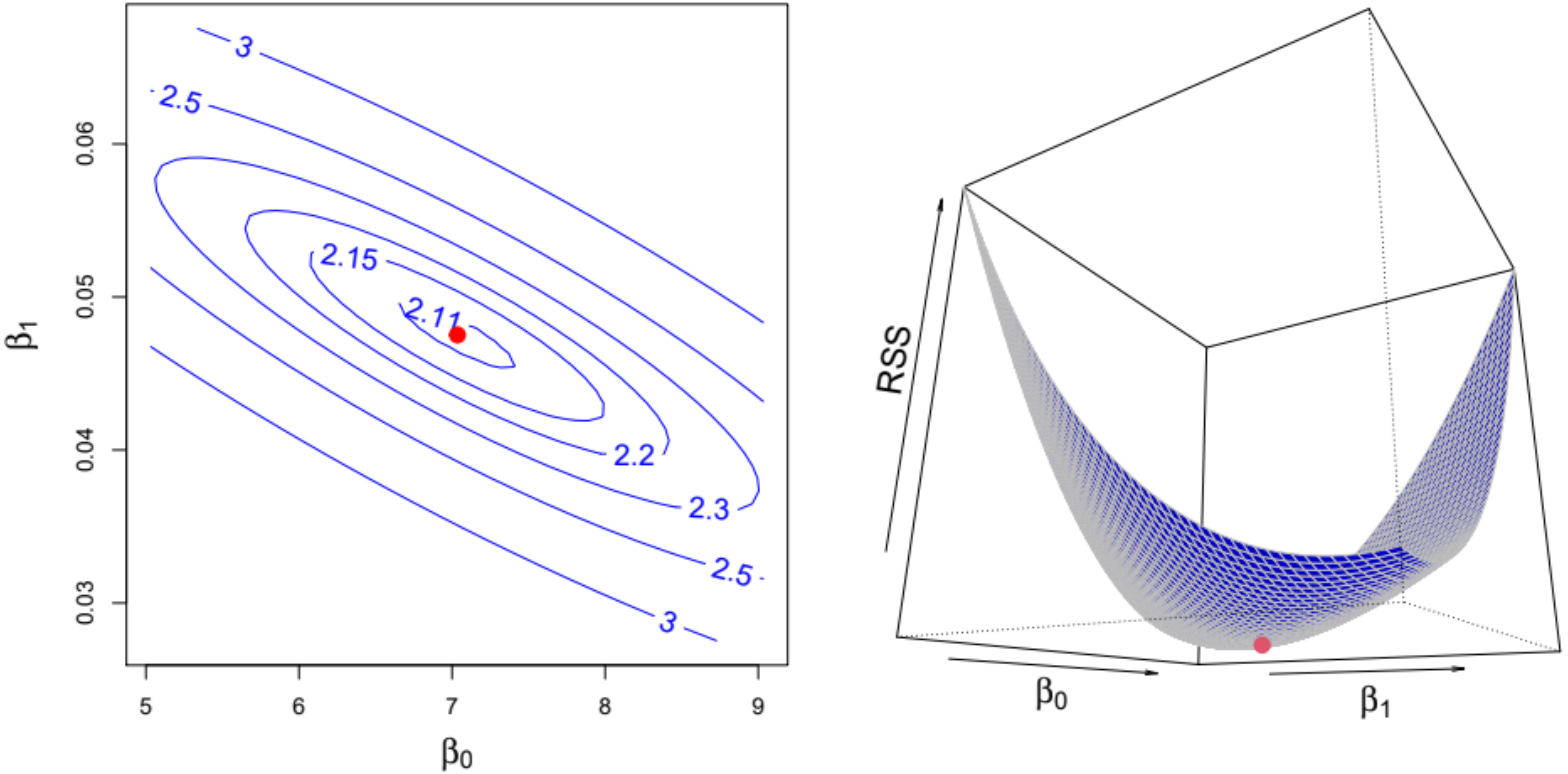
- How do we find the values of w_0 and w_1 that minimize $C(w_0, w_1)$?
- Because $\mathscr{H}$ is simple, we can optimize directly with calculus!
- Solutions:

$$w_0 = \frac{1}{n}\sum_{i=1}^{n} y_i - w_1 x_i$$

$$w_1 = \frac{n\sum_i x_i y_i - \left(\sum_i x_i\right)\left(\sum_i y_i\right)}{n\sum_i x_i^2 - \left(\sum x_i\right)^2}$$

- Solution is unique because the cost function is convex in the parameters

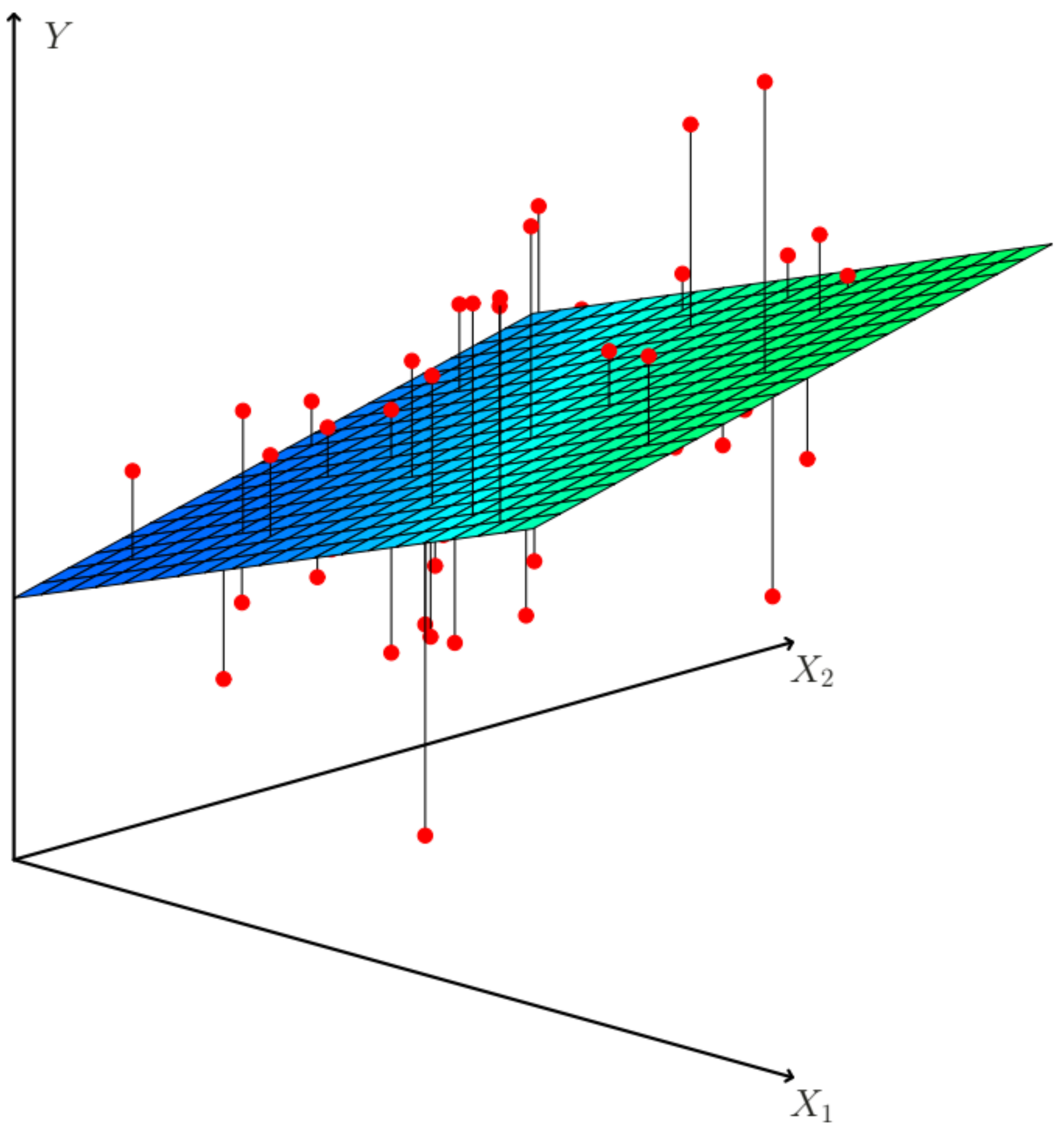

Loss
w_1
w_0



3
2.5
2.15
2.11
2.2
2.3
2.5
3
0.06
0.05
0.04
0.03
β1
5
6
7
8
9
β0
RSS
β0
β1

Linear regression with multiple features

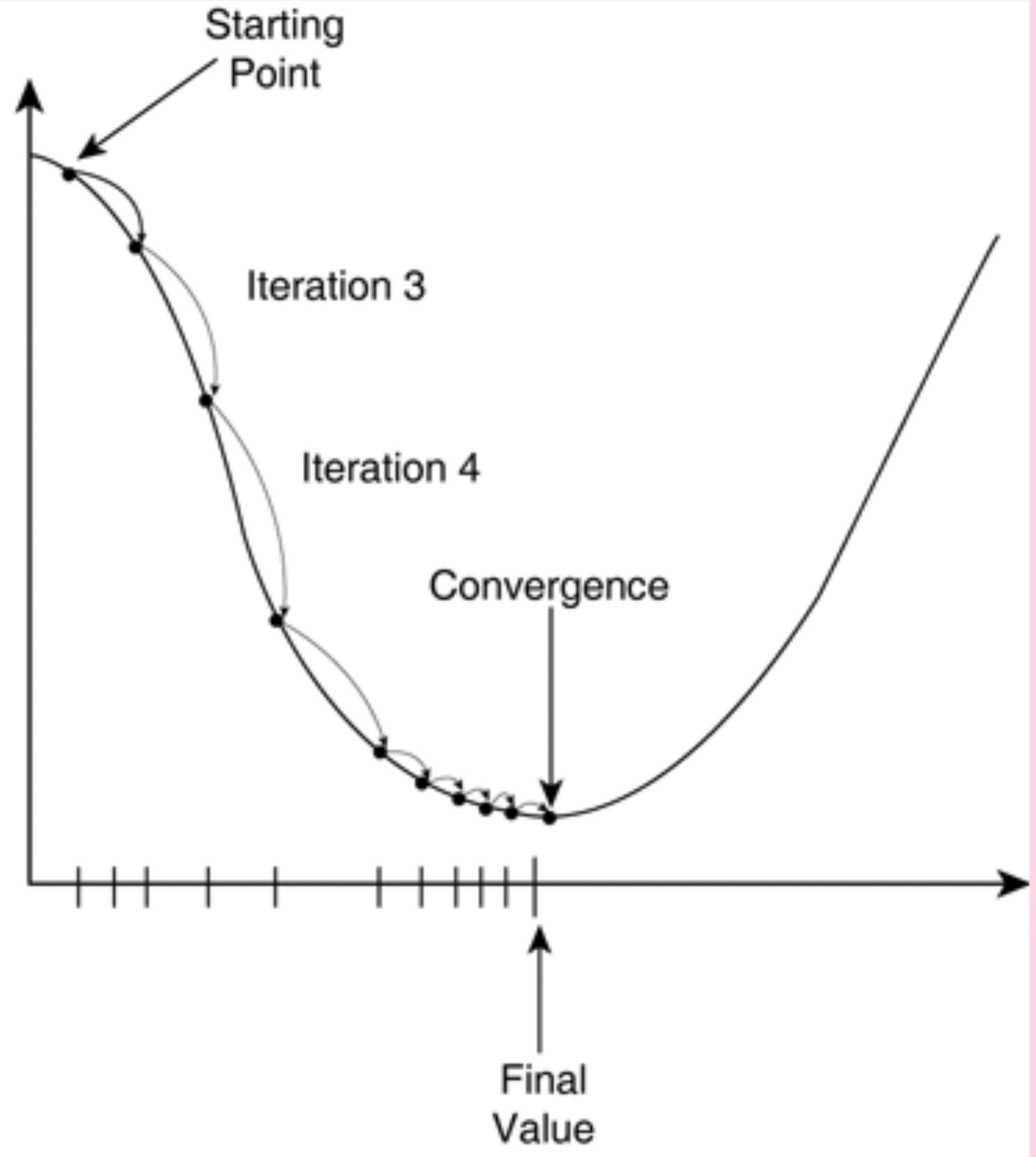
- Every data point is now a *vector* of features: $\mathbf{x} = (x_1, x_2, \ldots, x_p)$
- Notation: use superscripts to index into training set: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$
- **Hypothesis class:** linear models from feature tuples $(x_1, \ldots, x_p)$ to real numbers: $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0 = w_0 + \sum_{j=1}^{p} w_j x_j$
- Can use *multivariable* calculus to derive analytical solution: $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ for data matrix $\mathbf{X}$ and label vector $\mathbf{y}$
- Candidate hypotheses are hyperplanes in $(p+1)$-dimensional space



Y
X_2
X_1

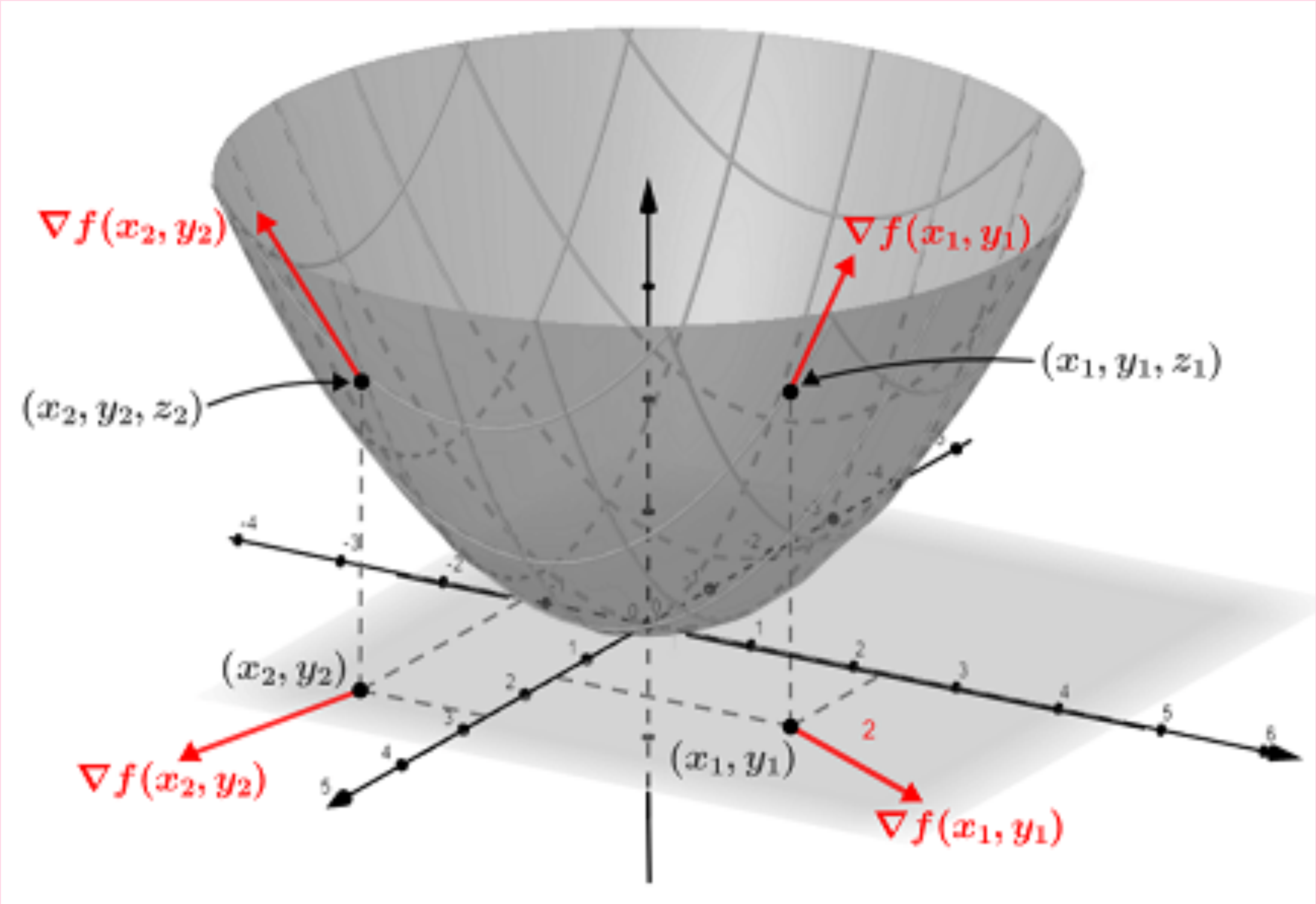
Alternative optimizer: gradient descent

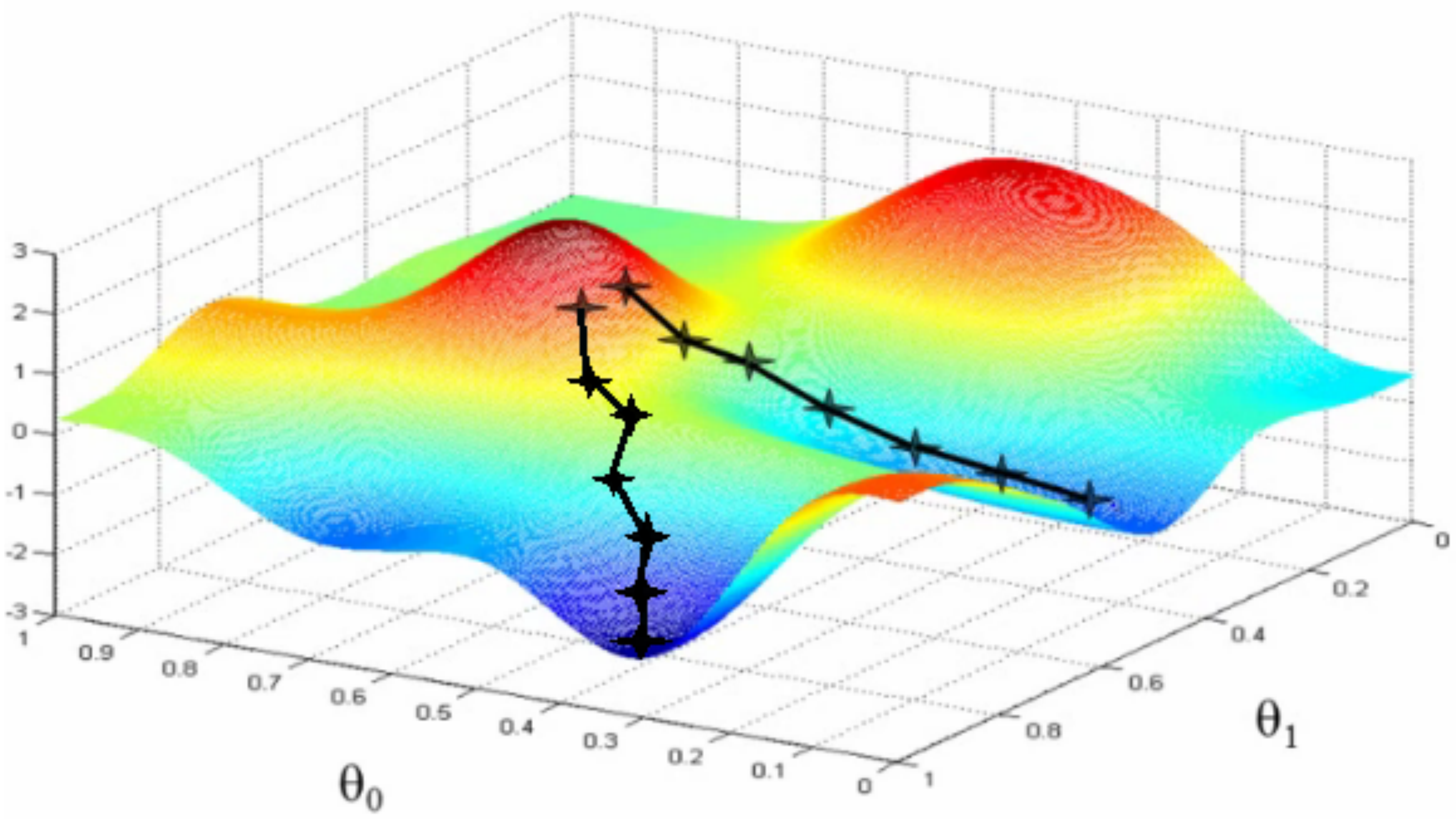
- **Goal:** pick parameters **w** to minimize cost function $C(\mathbf{w})$
- Basic idea: roll downhill
- How do you know which way is downhill?
- In one dimension, use the **derivative**
  - Positive derivative: step left
  - Negative derivative: step right
  - Step size should be proportional to magnitude of derivative
- Rule: $w \leftarrow w - \eta \frac{d}{dw} C(w)$



Gradient descent in several dimensions

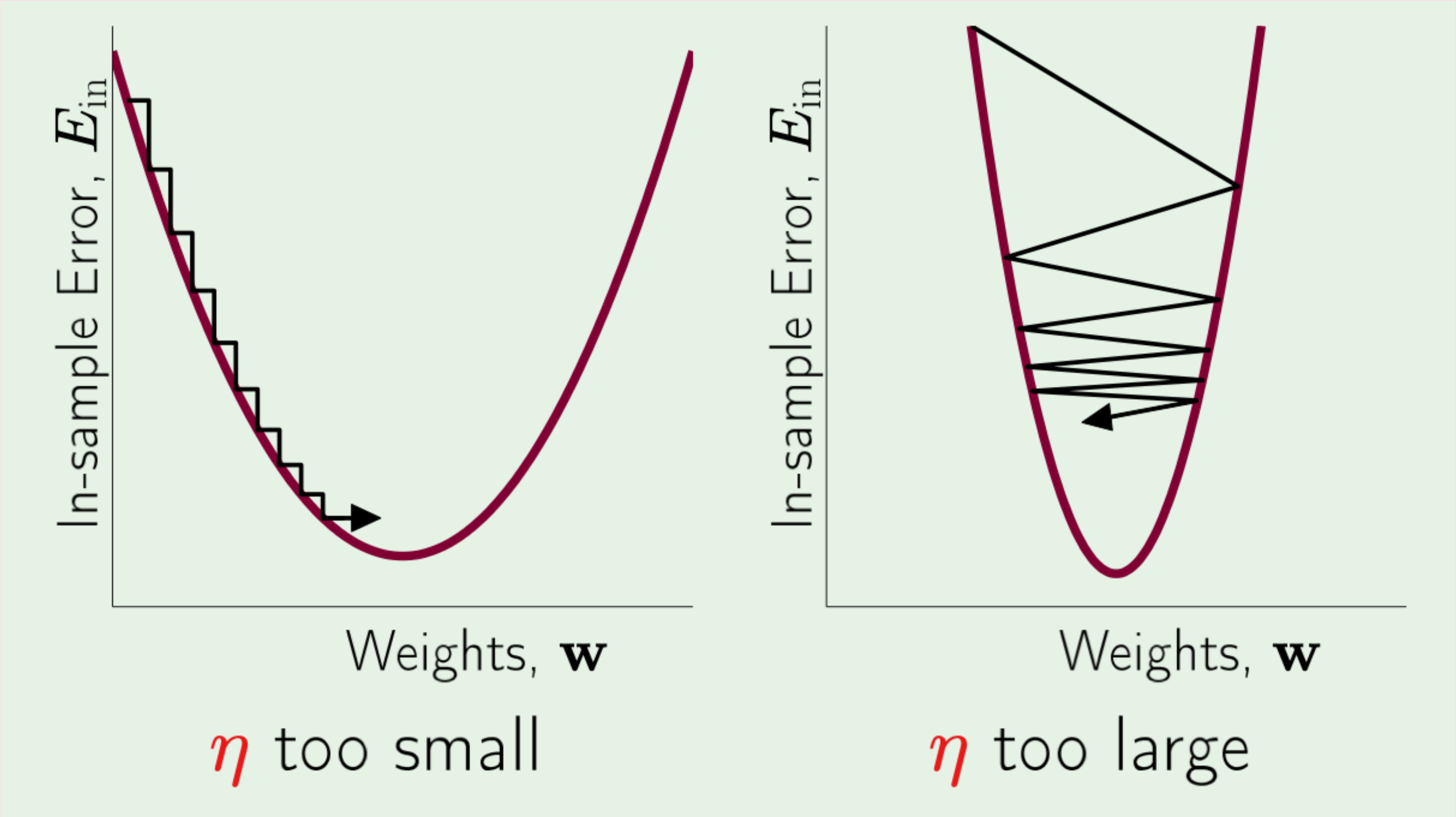
- In several dimensions, use the **gradient**
- $\nabla_{\mathbf{w}} C(\mathbf{w}) = \left(\frac{\partial}{\partial w_1} C(\mathbf{w}), \ldots, \frac{\partial}{\partial w_p} C(\mathbf{w}) \right)$
- The gradient is a vector that points in the direction of steepest ascent, with magnitude proportional to the slope in that direction
- For minimization, need to go the opposite direction of the gradient
- Rule: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} C(\mathbf{w})$




3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1

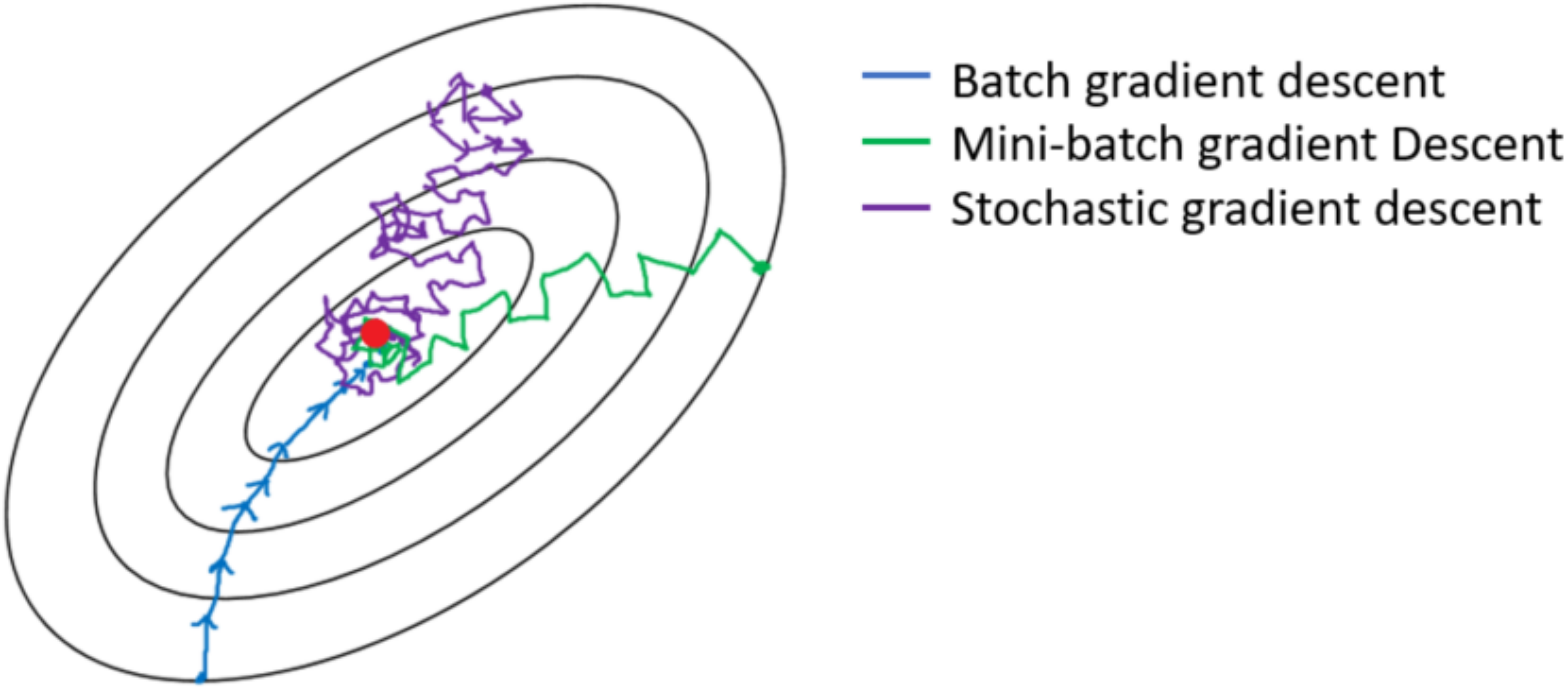
Problems with gradient descent

- What can go wrong with rolling downhill?
- Might converge to a *local* minimum instead of a *global* minimum
- Wrong step size η can lead to slow/no convergence
- Evaluating $\nabla_{\mathbf{w}} C(\mathbf{w})$ requires a pass through the entire training dataset; this is slow if the dataset is large



Stochastic gradient descent

- $C(\mathbf{w}) = \sum_{i=1}^{n} c\left(h_{\mathbf{w}}(\mathbf{x}^{(i)}), y^{(i)}\right)$
- Evaluating $\nabla_{\mathbf{w}} C(\mathbf{w})$ requires a pass through the entire training dataset
- Training dataset might contain $n =$ millions of examples
- Instead of using all n examples, take a sample of size b (typically $b \approx 50$)
- This sample is called a minibatch
- We can estimate $\nabla_{\mathbf{w}} C(\mathbf{w})$ using just the examples in the minibatch


Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent