Linear Regression

Goal

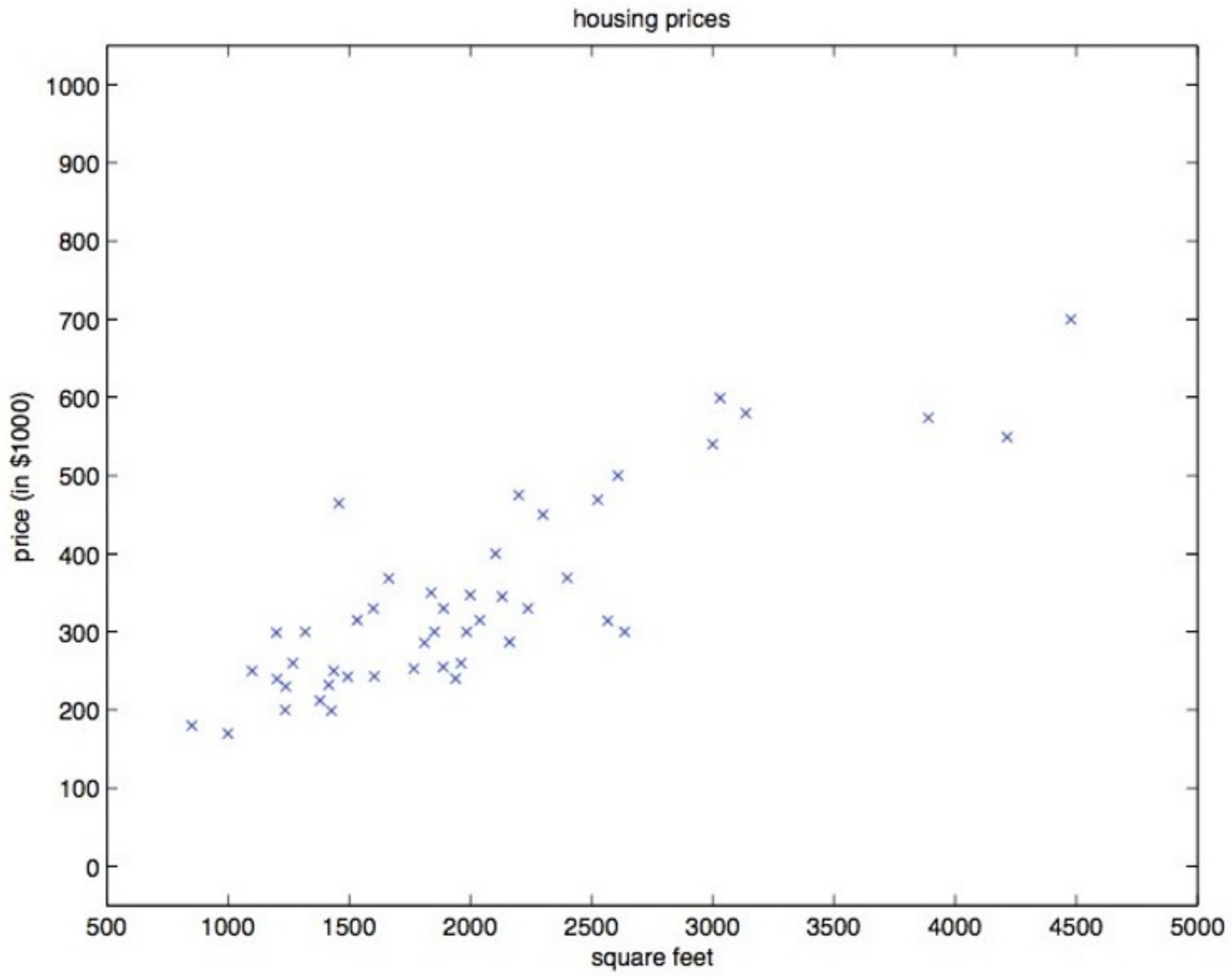
- Modeling the relationship between:
  - continuous input $X \in \mathbb{R}^d$
  - continuous output $Y \in \mathbb{R}^m$
    - Commonly denoted by “regression”

- Looks like a very general problem
  - We’ll make heavy simplifying assumptions

Example: House price prediction

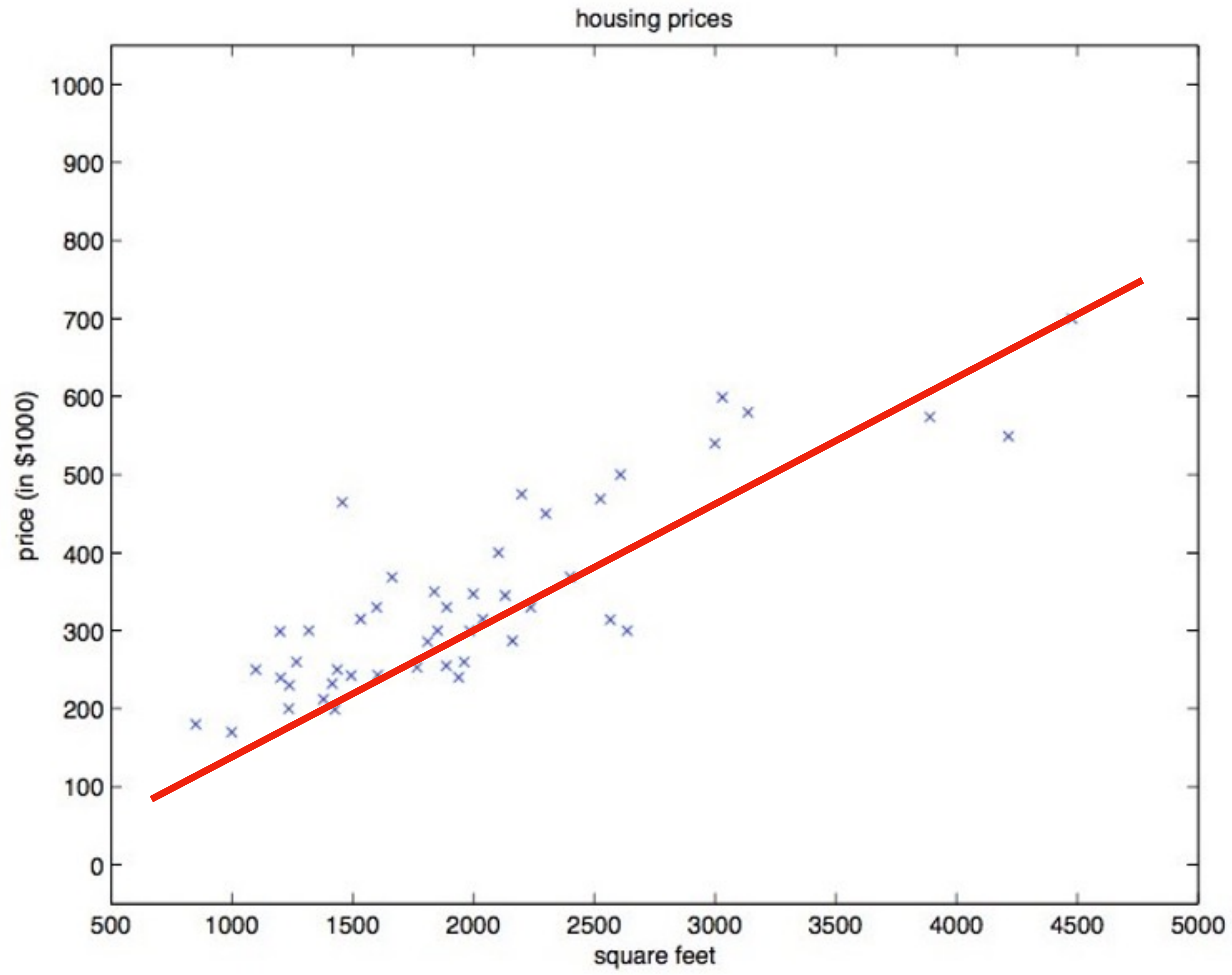
- Given the living area of a house, find the right estimate $f(\text{area}) = \text{price}$

Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



Model

- We use a **linear model** (or "affine," to be more precise)
  - $f(x) = g(x) + b$, where $g(\,\cdot\,)$ satisfies $g(cx) = c \cdot g(x) \quad \forall c \in \mathbb{R}$



Model

- For $x \in \mathbb{R}, y \in \mathbb{R}$:

$$f(x) = wx + b, \qquad w \in \mathbb{R}, b \in \mathbb{R}$$

- For $\mathbf{x} \in \mathbb{R}^d, y \in \mathbb{R}$

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \quad \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

- For $\mathbf{x} \in \mathbb{R}^d, y \in \mathbb{R}^m$

$$f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \quad \mathbf{W} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m$$

Model

Hypothesis space

- For $x \in \mathbb{R}, y \in \mathbb{R}$:

$$f(x) = wx + b, \qquad w \in \mathbb{R}, b \in \mathbb{R}$$

- For $\mathbf{x} \in \mathbb{R}^d, y \in \mathbb{R}$

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \qquad \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

- For $\mathbf{x} \in \mathbb{R}^d, y \in \mathbb{R}^m$

$$f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \qquad \mathbf{W} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m$$

Loss

- We will use the **squared ℓ_2 loss**

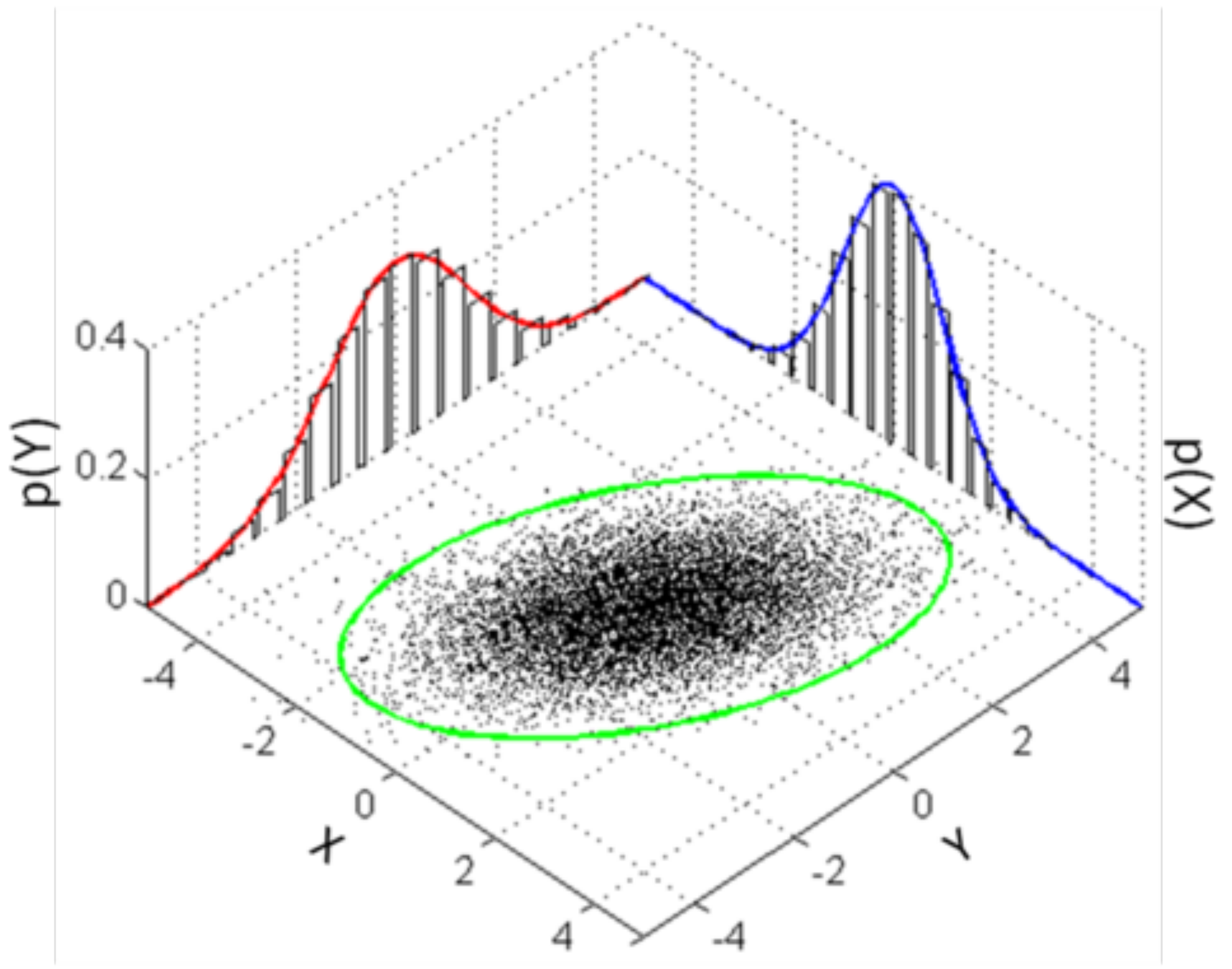
$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$$

- **Question.** Why the squared loss?
  - Easy to solve — quadratic function
  - Nice interpretation — Gaussian noise assumption (discussed later)

Loss

Note (for advanced readers).

- Recall that this loss function encourages learning $\eta(\mathbf{x}) := \mathbb{E}[\mathbf{y} \,|\, \mathbf{x}]$
  - However, as we use the linear model, we won't learn $\eta(\,\cdot\,)$ unless this is indeed a linear function.
- Fun fact: If $\mathbf{x}, \mathbf{y}$ are "jointly Gaussian," $\eta(\mathbf{x})$ is a linear function
  - Thus no "underfitting" in such case



ERM Objective

- Suppose that we are given a dataset

$$D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$$

- In linear regression, we solve the empirical risk minimization:

$$\min_{\mathbf{W},\mathbf{b}} \frac{1}{n} \sum_{i=1}^n \left(\mathbf{y}_i - (\mathbf{W}\mathbf{x}_i + \mathbf{b})\right)^2$$

- **Question.** How do we solve this optimization?
    - Analytic
    - Heuristic (Gradient Descent)

Training

Training

- Let us begin with a **1D, bias-free** case
    - That is, we consider the predictors

$$f(x) = wx, \qquad w, x \in \mathbb{R}$$

- Then, the ERM objective becomes a **quadratic function of** w:

$$\begin{aligned} J(w) &:= \frac{1}{n}\sum_{i=1}^{n}\left(y_i - wx_i\right)^2 \\ &= w^2\left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) + w\left(-\frac{2}{n}\sum_{i=1}^{n} x_i y_i\right) + y_i^2 \end{aligned}$$

Training

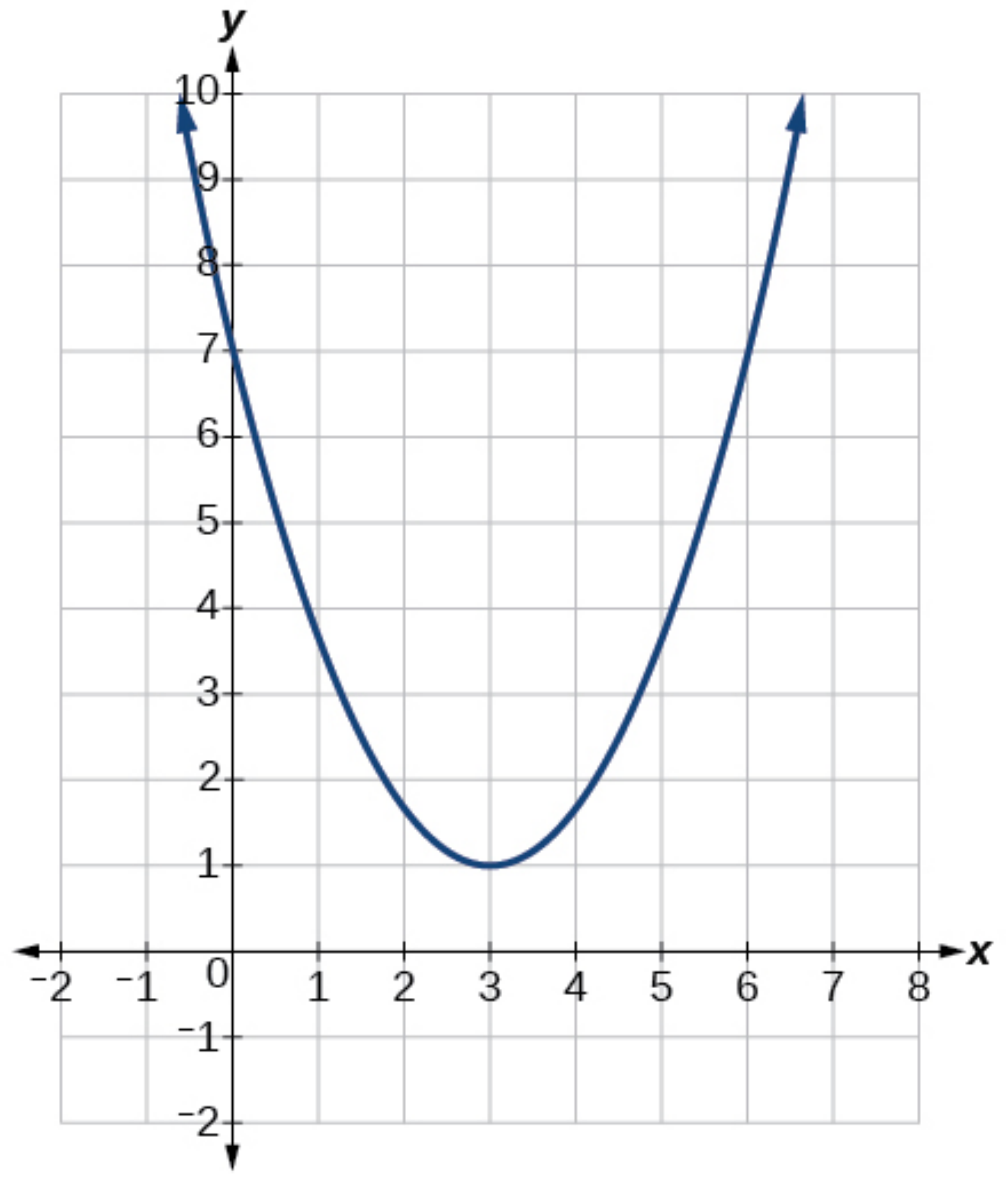
$$J(w) = w^2\left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) + w\left(-\frac{2}{n}\sum_{i=1}^{n} x_i y_i\right) + y_i^2$$

- This is a quadratic function with a **positive leading coefficient**:
  - Search space is unrestricted—the minimizer is the critical point

$$\frac{\partial}{\partial w} J(w) = 0 \Leftrightarrow w\left(\sum_{i=1}^{n} x_i^2\right) = \left(\sum_{i=1}^{n} x_i y_i\right)$$

  - Thus, we have

$$w^* = \left(\sum_{i=1}^{n} x_i y_i\right) / \left(\sum_{i=1}^{n} x_i^2\right)$$



Training

- This was an **exceptional case** where we have an **analytical solution**

- We won't always be this lucky –
  - What if our loss was $\ell(\hat{y}, y) = (y - \hat{y})^6$? (c.f. Abel-Ruffini Theorem)
  - What is our model was much more complicated?

- That's why we have heuristic methods as well
  - We'll see later today

Multivariate case

- Now consider a slightly more general case, with $\mathbf{x} \in \mathbb{R}^d$, $y \in \mathbb{R}$
    - We'll start to see why we need linear algebra & vector calculus

- Then, the ERM objective will be:

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \mathbf{w}^\top \mathbf{x}_i + b\right)^2$$

- Things start to look a bit messy
    - We'll first simplify using stacked notations

Multivariate case

- First, we **stack parameters** by using shorthands

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \theta = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

- Then, our ERM objective becomes

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^\top \bar{\mathbf{x}}_i)^2$$

Multivariate case

- Second, we **stack data** by using shorthands

$$\mathbf{X} = \begin{bmatrix} \tilde{\mathbf{x}}_1^\top \\ \cdots \\ \tilde{\mathbf{x}}_n^\top \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ \cdots \\ y_n \end{bmatrix}$$

- Then, our ERM objective becomes

$$J(\theta) = \frac{1}{n} \| \mathbf{y} - \mathbf{X}\theta \|^2$$

Multivariate case

$$J(\theta) = \frac{1}{n} \parallel \mathbf{y} - \mathbf{X}\theta \parallel^2$$

- Now we examine the critical point condition:

$$\begin{aligned}
\nabla J(\theta) &= \frac{1}{n}\nabla\left((\mathbf{y} - \mathbf{X}\theta)^\top(\mathbf{y} - \mathbf{X}\theta)\right) \\
&= \frac{1}{n}\nabla\left(\mathbf{y}^\top\mathbf{y} + \theta^\top\mathbf{X}^\top\mathbf{X}\theta - 2\mathbf{y}^\top\mathbf{X}\theta\right) \\
&= \frac{1}{n}\left(2\theta^\top\mathbf{X}^\top\mathbf{X} - 2\mathbf{y}^\top\mathbf{X}\right) = 0
\end{aligned}$$

Multivariate case

- Thus, the critical point condition is:

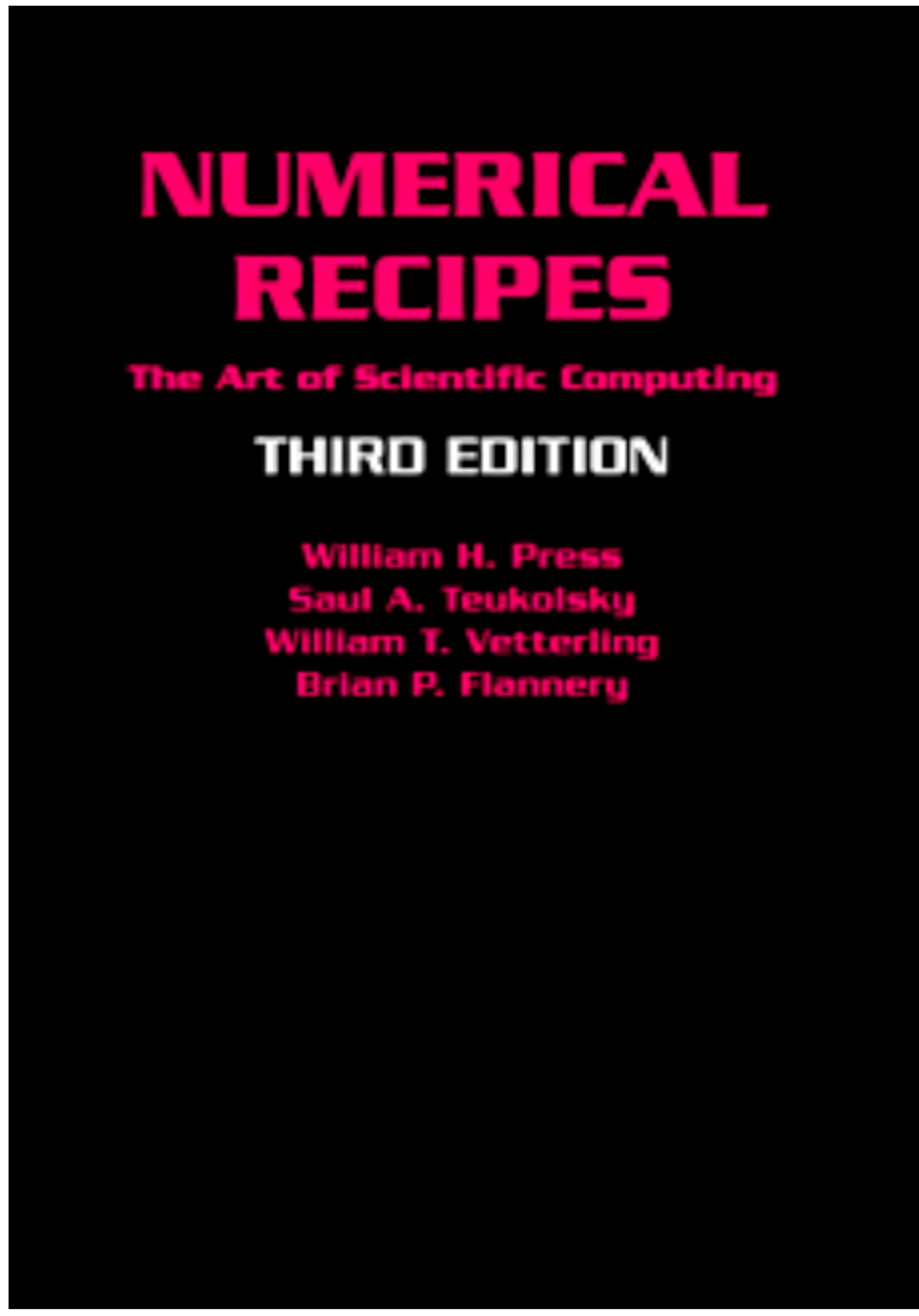
$$\mathbf{X}^\top \mathbf{X}\theta = \mathbf{X}^\top \mathbf{y}$$

- Thankfully, this problem has a rather classic form of

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

with known $\mathbf{A}$ and $\mathbf{b}$

  - Has been a studied for a long time
  - Techniques introduced in basic linear algebra
    - Mathematics of ML: https://mml-book.github.io/
    - Numerical Recipes (advanced)



Multivariate case

$$\mathbf{X}^\top \mathbf{X} \theta = \mathbf{X}^\top \mathbf{y}$$

- If $\mathbf{X}^\top \mathbf{X}$ is invertible, we can simply solve by inverting

$$\theta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

  - Unique solution guaranteed — no headaches

- Sadly, not always invertible
  - If $n < d + 1$, it is never invertible — because $\mathbf{X}^\top \mathbf{X}$ is at most rank $\min\{n, d+1\}$ (too few data)
  - Depends on data $\mathbf{X}$, which is random

Multivariate case

$$\mathbf{X}^\top \mathbf{X} \theta = \mathbf{X}^\top \mathbf{y}$$

- If $\mathbf{X}^\top \mathbf{X}$ is not invertible, we'll have **infinitely many solution**
  - Called "underdetermined"
  - Still, we know that any θ that satisfies above will be a global minima

- To get one of these solutions, we can use the QR decomposition:
  - For those who don't remember, let's do a quick recap
  - Quick fact: There are other methods, but QR decomposition is known to be more numerically stable

Recap: QR Decomposition

Recap: QR Decomposition

- Suppose that we have a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$
  - Further assume that $m \geq n$
- Then, QR decomposition decomposes the matrix as

$$\mathbf{A} = \mathbf{QR}$$

  - $\mathbf{Q} \in \mathbb{R}^{m \times m}$ is a unitary matrix (i.e., $\mathbf{Q}^\top = \mathbf{Q}^{-1}$)
  - $\mathbf{R} \in \mathbb{R}^{m \times n}$ is an upper triangular matrix

$$\mathbf{A} = \begin{bmatrix} | & \cdots & | \\ \mathbf{e}_1 & \cdots & \mathbf{e}_m \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ & & \cdots & \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

($\mathbf{e}_i$ are orthonormal — orthogonal to each other, and $||\mathbf{e}_i||_2 = 1$)

Recap: QR Decomposition

$$\mathbf{A} = \begin{bmatrix} | & \cdots & | \\ \mathbf{e}_1 & \cdots & \mathbf{e}_m \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ & & \cdots & \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

- **Idea.** Take a look each column of $\mathbf{A}$:

$$\mathbf{a}_1 = \begin{bmatrix} | & \cdots & | \\ \mathbf{e}_1 & \cdots & \mathbf{e}_m \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} r_{11} \\ 0 \\ 0 \\ \cdots \end{bmatrix}, \quad \mathbf{a}_2 = \begin{bmatrix} | & \cdots & | \\ \mathbf{e}_1 & \cdots & \mathbf{e}_m \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} r_{12} \\ r_{22} \\ 0 \\ \cdots \end{bmatrix}, \quad \cdots$$

  - That means we're breaking down $\mathbf{a}_1 = r_{11}\mathbf{e}_1$

$$\mathbf{a}_2 = r_{12}\mathbf{e}_1 + r_{22}\mathbf{e}_2$$

(…)

Recap: QR Decomposition

$$\mathbf{a}_1 = r_{11}\mathbf{e}_1, \quad \mathbf{a}_2 = r_{12}\mathbf{e}_1 + r_{22}\mathbf{e}_2, \quad \cdots$$

- Realizing this, our algorithm becomes straightforward
  - **Gram-Schmidt process**

- **Step 1**. Make $\mathbf{e}_1$ by normalizing $\mathbf{a}_1$

$$\mathbf{e}_1 = \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|_2}, \quad r_{11} = \|\mathbf{a}_1\|_2$$

Recap: QR Decomposition

$$\mathbf{a}_1 = r_{11}\mathbf{e}_1, \quad \mathbf{a}_2 = r_{12}\mathbf{e}_1 + r_{22}\mathbf{e}_2, \quad \cdots$$

- Realizing this, our algorithm becomes straightforward
  - **Gram-Schmidt process**

- Step 1. Make $\mathbf{e}_1$ by normalizing $\mathbf{a}_1$
- **Step 2.** Make $\mathbf{e}_2$ by (1) subtracting the $\mathbf{a}_1$ direction, and (2) normalizing the remainder

$$r_{12} = \mathbf{a}_2^\top \mathbf{e}_1, \quad \mathbf{e}_2 = \frac{\mathbf{a}_2 - r_{12}\mathbf{e}_1}{\|\mathbf{a}_2 - r_{12}\mathbf{e}_1\|_2}, \quad r_{22} = \|\mathbf{a}_2 - r_{12}\mathbf{e}_1\|_2$$

Recap: QR Decomposition

$$\mathbf{a}_1 = r_{11}\mathbf{e}_1, \quad \mathbf{a}_2 = r_{12}\mathbf{e}_1 + r_{22}\mathbf{e}_2, \quad \cdots$$

- Realizing this, our algorithm becomes straightforward
  - **Gram-Schmidt process**

- **Step 1.** Make $\mathbf{e}_1$ by normalizing $\mathbf{a}_1$
- **Step 2.** Make $\mathbf{e}_2$ by (1) subtracting the $\mathbf{a}_1$ direction, and
  (2) normalizing the remainder

- **Step 3.** Repeat!

Recap: Pseudoinverse

- Using QR decomposition, we can get a **Moore-Penrose pseudoinverse**
    - That is, a matrix $\mathbf{A}^\dagger \in \mathbb{R}^{n \times m}$ satisfying:
        - $\mathbf{A}\mathbf{A}^\dagger\mathbf{A} = \mathbf{A}, \qquad \mathbf{A}^\dagger\mathbf{A}\mathbf{A}^\dagger = \mathbf{A}^\dagger$
        - $(\mathbf{A}\mathbf{A}^\dagger)^\top = \mathbf{A}\mathbf{A}^\dagger, \quad (\mathbf{A}^\dagger\mathbf{A})^\top = \mathbf{A}^\dagger\mathbf{A}$

- Using the QR decomposition, you can compute the pseudoinverse as:

$$\mathbf{A}^\dagger = \mathbf{R}^{-1}\mathbf{Q}^\top$$

    - If we have $m \leq n$ or rank-deficient case — consult your linear algebra textbook!

</QR>

Multivariate case

$$\mathbf{X}^\top \mathbf{X}\theta = \mathbf{X}^\top \mathbf{y}$$

- To get one solution, we can use the pseudoinverse:

$$\hat{\theta} = (\mathbf{X}^\top \mathbf{X})^\dagger \mathbf{X}^\top \mathbf{y}$$

  - Remarkably, this solution happens to be a **minimum ℓ_2 norm solution** among all θ that satisfies $\mathbf{X}^\top \mathbf{X}\theta = \mathbf{X}^\top \mathbf{y}$.

- **Fun exercise.** Count the computational cost of solving pseudoinverse:
  - number of FLOPs
  - memory cost

(Hint: Depends on the order of computation!)

Non-analytical solution: Gradient descent

Gradient Descent

- Let’s explore another way to solve the linear regression
    - A heuristic method, called gradient descent

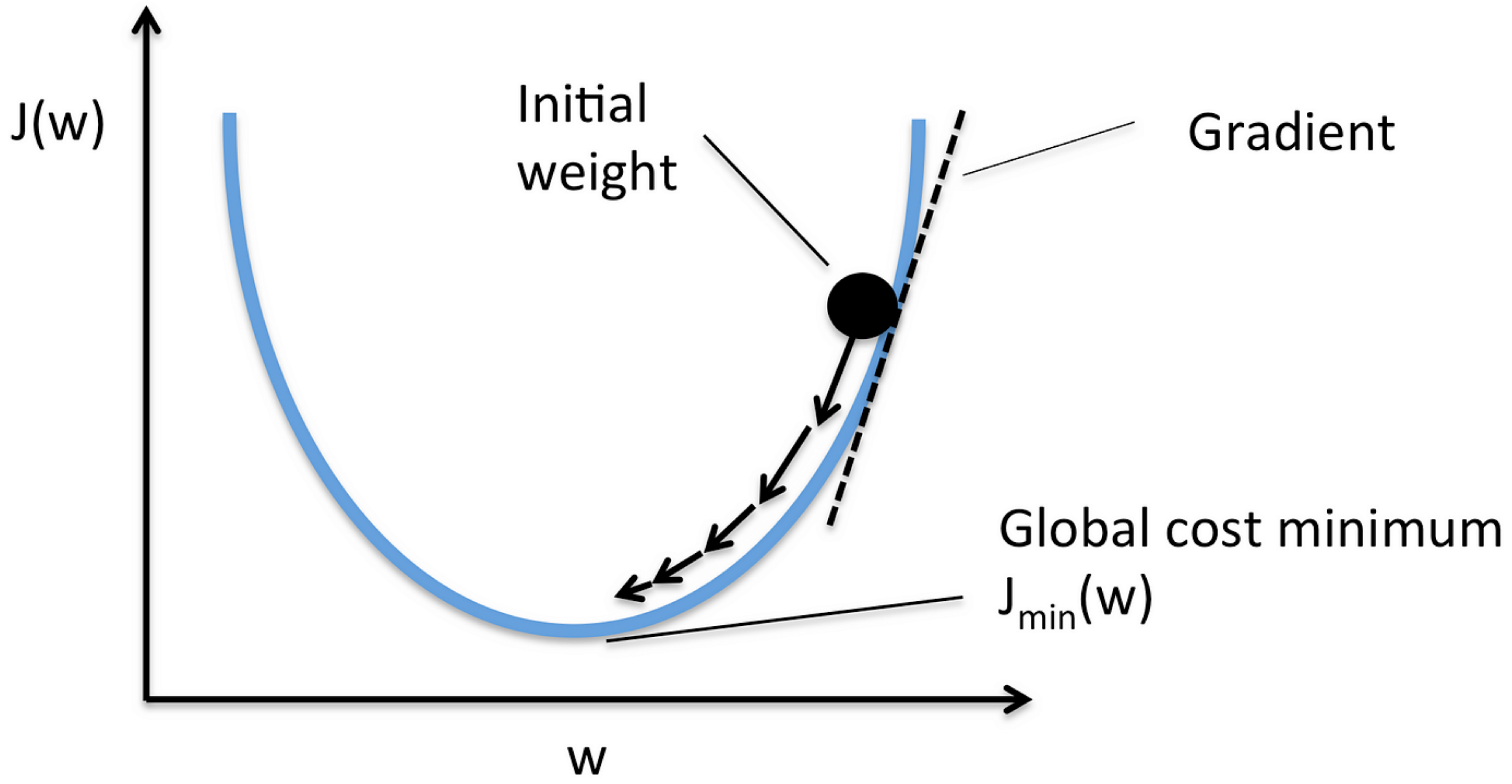
- **Intuition.** To minimize some function, repeat taking steps toward the downward direction

Gradient Descent

- A bit more formally – to minimize $J(\theta)$:
  - Randomly pick an initial parameter $\theta^{(0)}$
  - Repeat making a small update toward **negative gradient** direction

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta J(\theta^{(t)})$$

  - η: step size

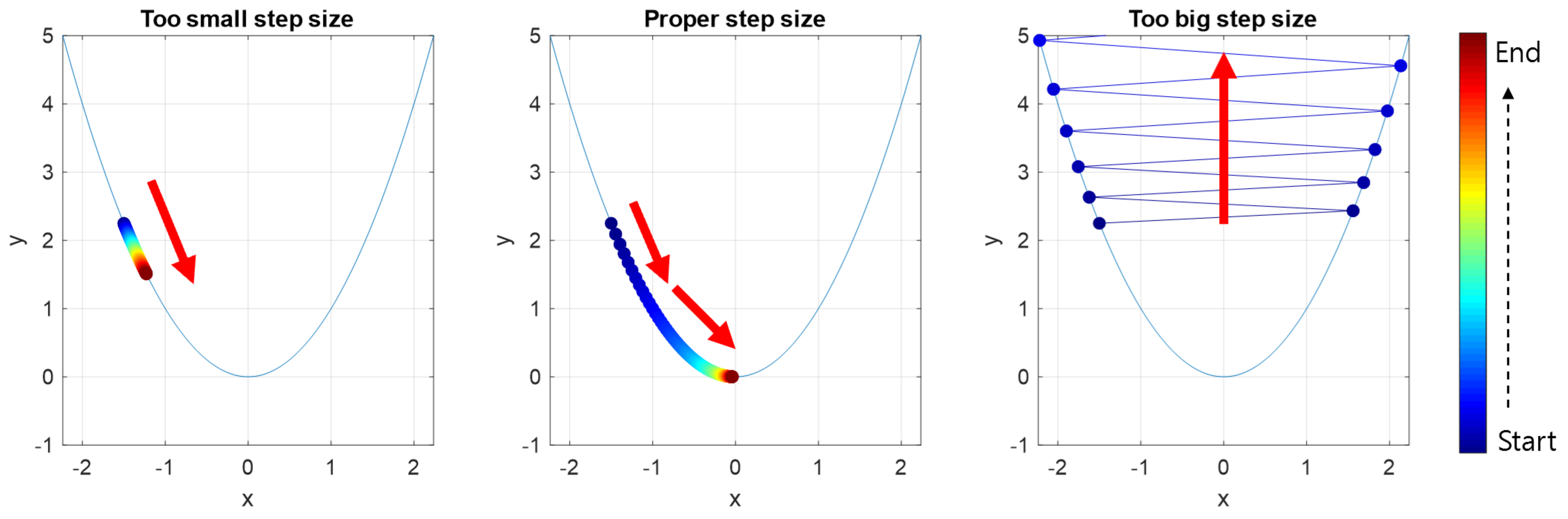


Gradient Descent, for Linear Regression

- For linear regression, the iterative update becomes:

$$\theta \leftarrow \theta - \frac{2\eta}{n}\left(\mathbf{X}^\top \mathbf{X}\theta - \mathbf{X}^\top \mathbf{y}\right)$$

- Given **appropriate η**, it will approach a good-enough solution
  - If too big, will diverge
  - If too small, requires many steps



Selecting the right η

- Like η, some ML algorithms may contain parameters such that:
  - Have nontrivial influence on the success
  - Yet, not a subject of optimization during the training

- We call these **hyperparameters**
  - Can be tuned via trial-and-error
    - Use $\eta_1, \eta_2, \eta_3, \ldots$ to get models $\theta_1, \theta_2, \theta_3, \ldots$
    - Test the models on some **samples that are not used for training**
      - called **validation samples**
    - Select the best-working η

Selecting the right η

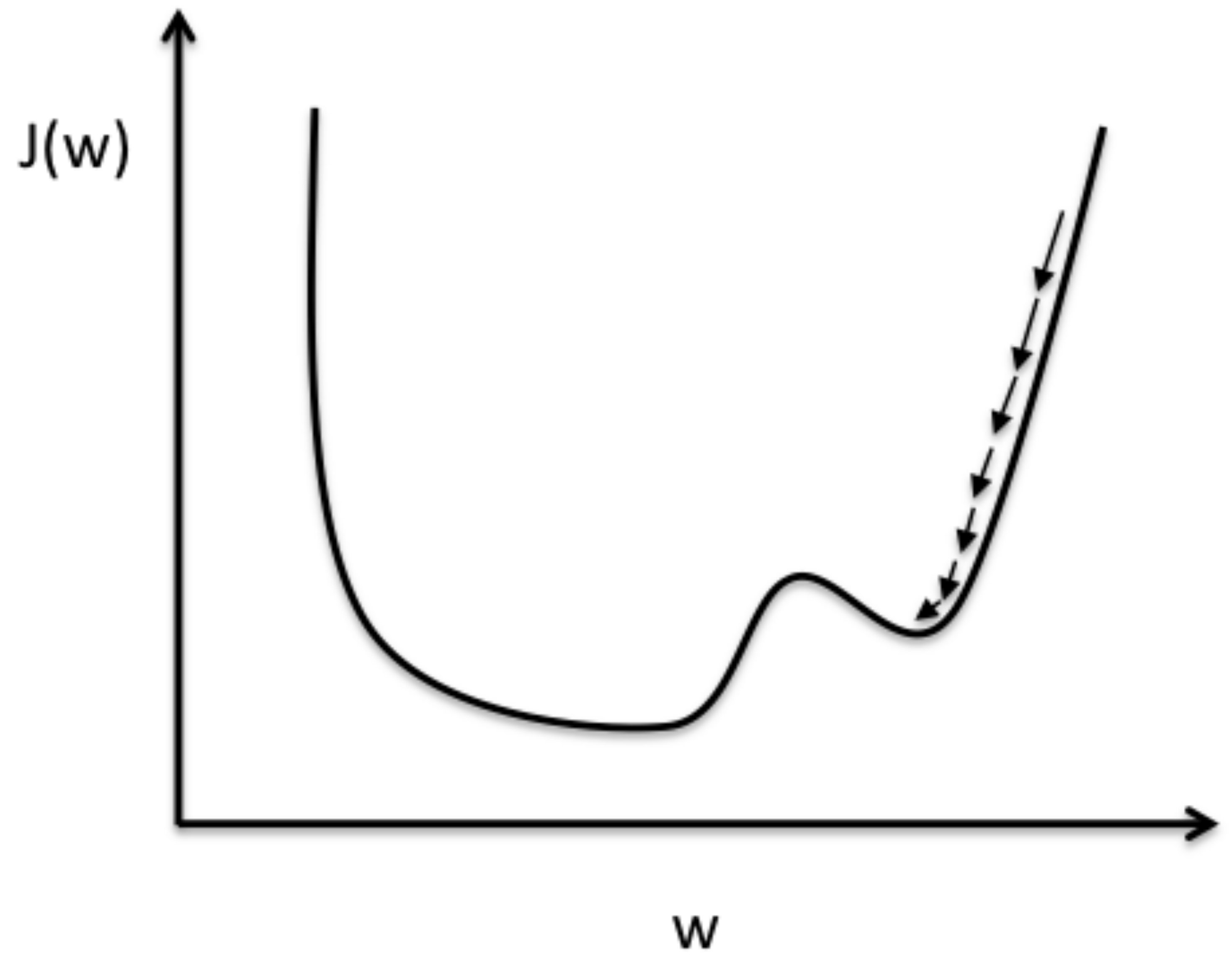
- Given some labeled dataset, we typically split it into **8:1:1** ratio for training, validation, test
  - Sometimes 7:1:2 — no fixed rule!

- Tuning these hyperparameters requires much computation and labor
  - AutoML algorithms have been proposed automated tuning

Remarks on Gradient Descent

Theoretical remarks

- No convergence guarantee in general
  - In simple cases, one can prove convergence
    - Often requires “scheduling” of η – e.g., diminishing it
- Worse, even at convergence, no guarantee that it will be optimal
  - Still handy in non-analytically-solvable cases
  - Works strangely well in deep learning



Remarks on Gradient Descent

Computational remarks

- Requires some computation, in general
  - Comparing with analytic solutions…
    - **Memory.** Typically GD is cheaper
    - **Compute.** Depends on #steps

- For linear regression, one can pre-compute and re-use — i.e., conduct

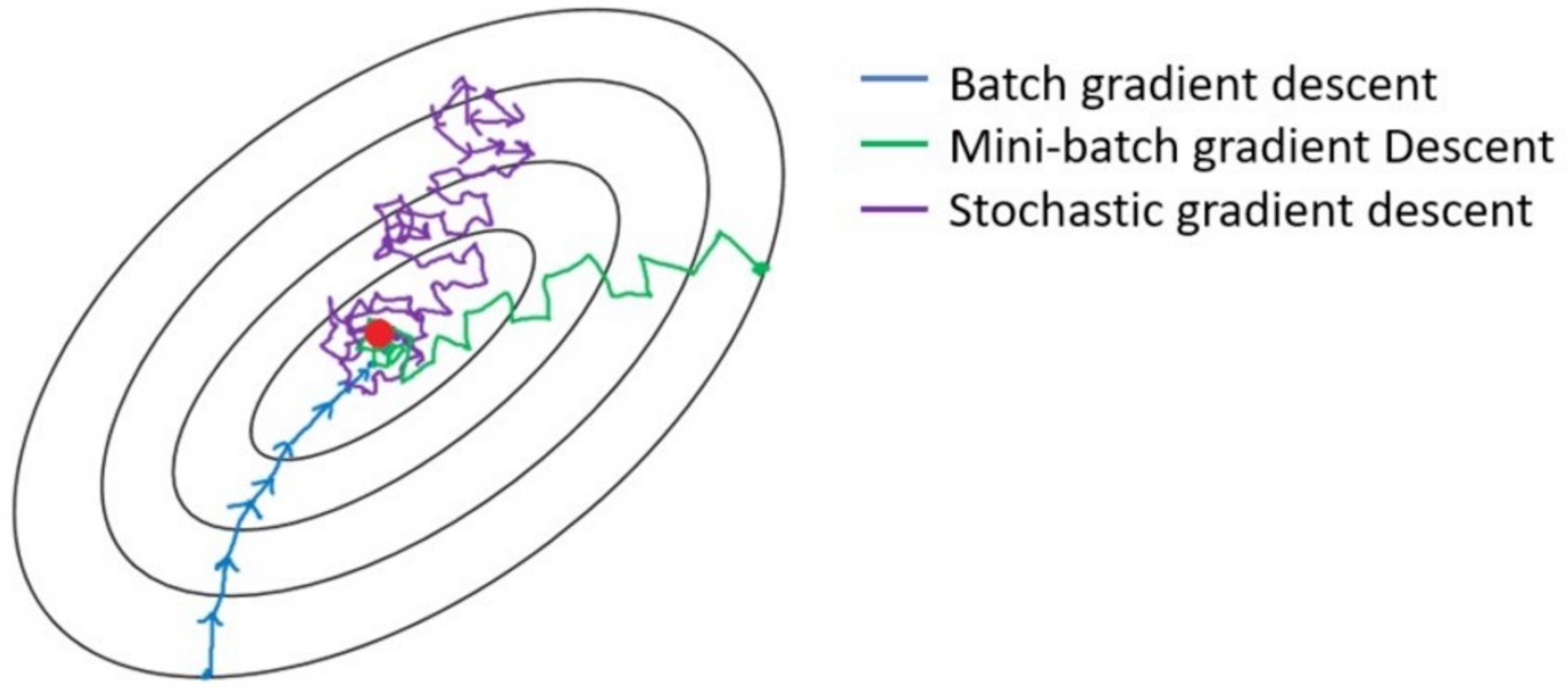
$$\theta \leftarrow (\mathbf{I} - \mathbf{A})\theta - \mathbf{b}$$

for $\mathbf{A} := \frac{\eta}{n}\mathbf{X}^\top\mathbf{X}$ and $\mathbf{b} := \frac{\eta}{n}\mathbf{X}^\top\mathbf{y}$

Remarks on Gradient Descent

Computational remarks

- To reduce the computational cost, we can use part of data only
  - Use a randomly drawn subset of k samples in each iteration $(k \ll n)$
    - Called mini-batch GD (or stochastic GD when $k = 1$)
    - Saves much RAM, and sometimes generalize better



Wrapping up

- Regression
- Linear model
- Squared loss
- Optimization
    - Analytical solution
    - Gradient descent

- **Next up.** Simple classifiers
    - Naïve Bayes, Nearest neighbors
    - Linear model

</lecture 3>