UCI University of California, Irvine

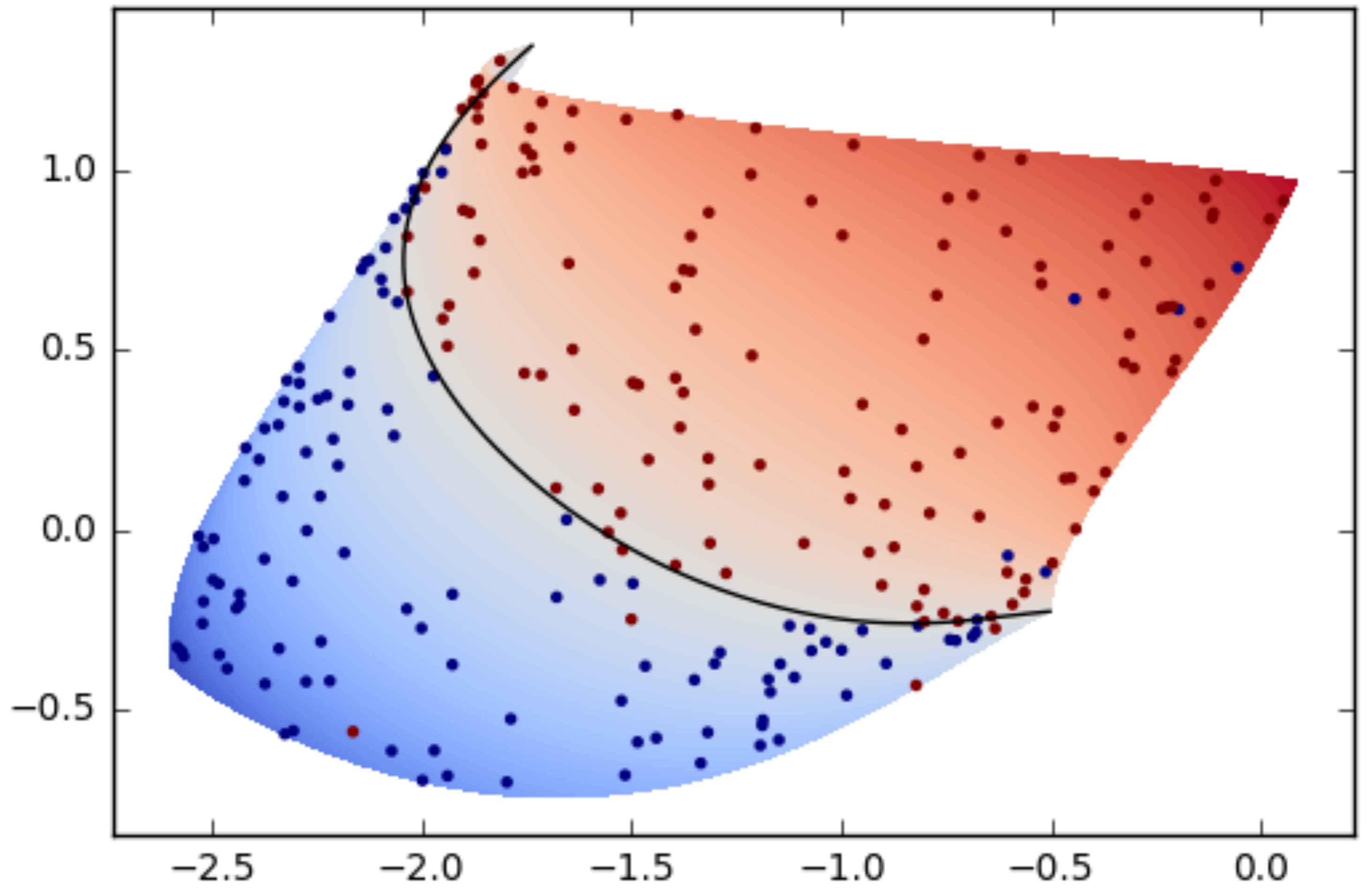
CS 273A: Machine Learning

Winter 2021

Lecture 5: Linear Regression (cont.)

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

All slides in this course adapted from Alex Ihler & Sameer Singh



Logistics

assignments

- Assignment 2 to be published soon

project

- Project guidelines to be published soon
- Team rosters due next Thursday, Jan 28

Today's lecture

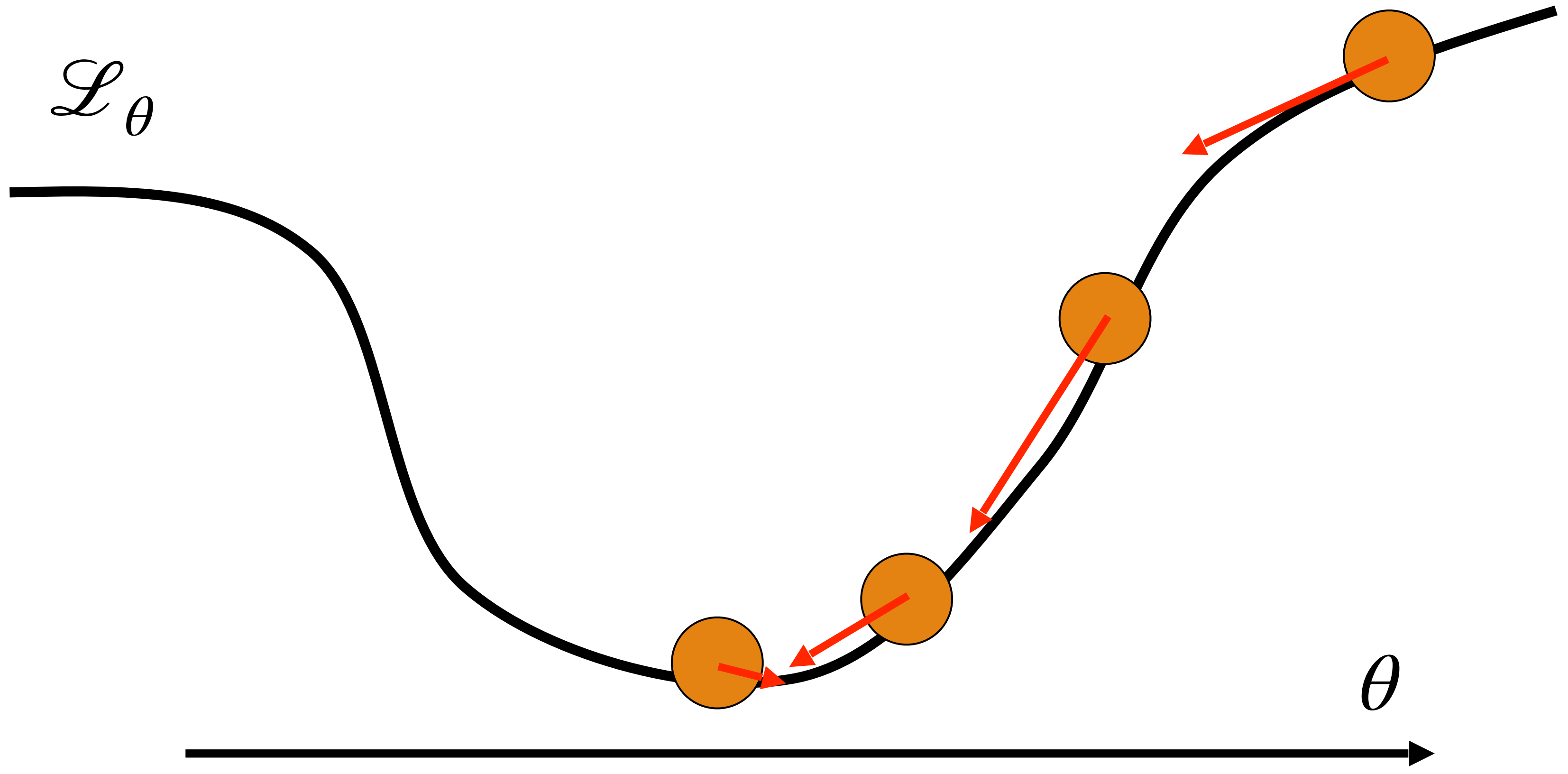
Stochastic Gradient Descent

Least Squares

Polynomial regression

Gradient Descent

- Initialize θ
- Do
    - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta$
- While $||\alpha \nabla_\theta \mathscr{L}_\theta|| \leq \epsilon$
- Learning rate: α
    - Can change in each iteration



Gradient for the MSE loss

- MSE: $\mathscr{L}_\theta = \frac{1}{m}\sum_j (\epsilon^{(j)})^2 = \frac{1}{m}\sum_j (y^{(j)} - \theta^\intercal x^{(j)})^2$

- $\partial_{\theta_i}\mathscr{L}_\theta = \frac{1}{m}\sum_j \partial_{\theta_i}(\epsilon^{(j)})^2 = \frac{1}{m}\sum_j 2\epsilon^{(j)}\partial_{\theta_i}\epsilon^{(j)}$

  - $\partial_{\theta_i}(y^{(j)} - \theta^\intercal x^{(j)}) = -\partial_{\theta_i}\theta_i x_i^{(j)} + 0$ in the other terms $= x_i^{(j)}$

  - $\partial_{\theta_i}\mathscr{L}_\theta = -\frac{2}{m}\sum_j \epsilon^{(j)} x_i^{(j)} = -\frac{2}{m}(y - \theta^\intercal X)X_i^\intercal$

- $\nabla_\theta \mathscr{L}_\theta = -\frac{2}{m}(y - \theta^\intercal X)X^\intercal$

error → $(y - \theta^\intercal X)$

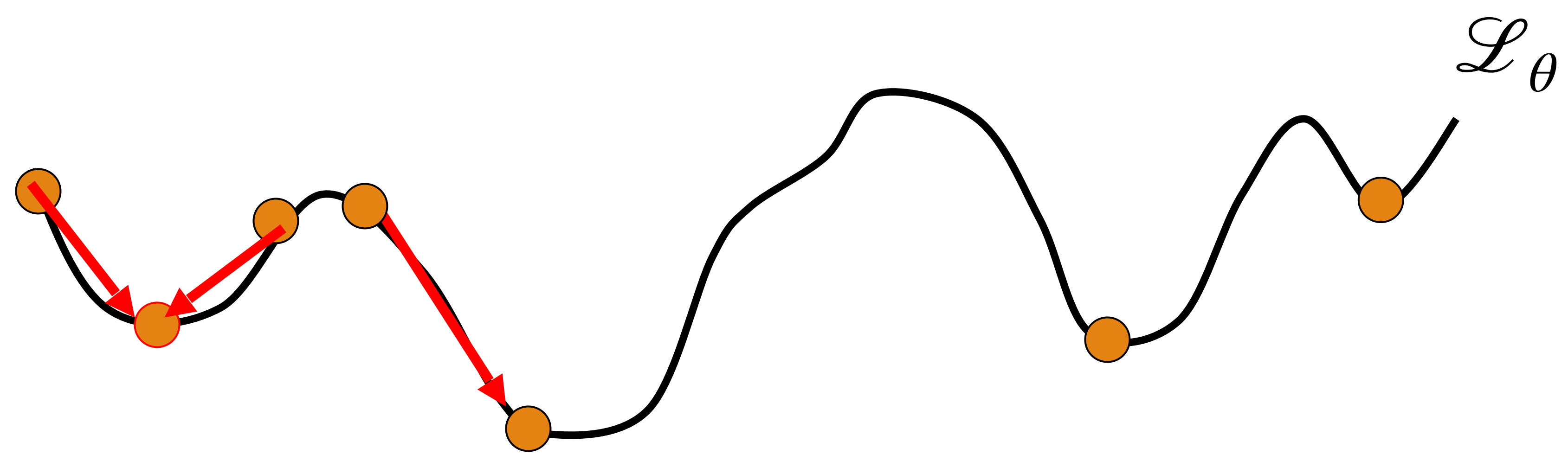
sensitivity to θ → $X^\intercal$

- Can also be seen directly from

$$\mathscr{L}_\theta = \frac{1}{m}(y - \theta^\intercal X)(y - \theta^\intercal X)^\intercal = \frac{1}{m}(\theta^\intercal X X^\intercal \theta - 2yX^\intercal\theta + yy^\intercal)$$

Gradient Descent — further considerations

- GD is a very general algorithm
  - We'll use it often
  - Much of the engine for recent advances in ML
- Issues:



  - Can get stuck in local minima
    - Worse — can get stuck in saddle points, $\nabla_\theta \mathscr{L}_\theta = 0$ with improvement direction
  - Can be slow to converge, sensitive to initialization
  - How to choose step size / learning rate?
    - Constant? 1/iteration? Line search? Newton's method?

Newton's method

- Given black-box $f(z)$, how to find a root $f(z) = 0$?
- Initialize some z
- Repeat:
  - Evaluate $f(z)$ and $\partial_z f(z)$ to find tangent to f at z: $f'(z') = (z' - z)\partial_z f(z) + f(z)$
  - Update z to the root of f': $z \leftarrow z - \dfrac{f(z)}{\partial_z f(z)}$
- Considerations:
  - May not converge, sometimes unstable
  - Usually converges quickly for nice, smooth, locally quadratic functions

Newton's method for gradient descent

- We want to find a (local) minimum $f(\theta) = \nabla_\theta \mathscr{L}_\theta = 0$
- Initialize some θ
- Repeat:
  - Evaluate gradient $g = \nabla_\theta \mathscr{L}_\theta$ and Hessian $H = \nabla^2_\theta \mathscr{L}_\theta$
  - Update $\theta \leftarrow \theta - H^{-1} g$
- Considerations:
  - Update step may be too large for highly non-convex losses
  - Computational complexity to invert H: $O(n^3)$

Gradient Descent: complexity

- Assume $\mathscr{L}_\theta(\mathscr{D}) = \frac{1}{m}\sum_j \ell_\theta(x^{(j)}, y^{(j)})$
  - MSE: $\ell_\theta(x, y) = (y - \theta^\intercal x)^2$
- Computing $\nabla_\theta \mathscr{L}_\theta = \frac{1}{m}\sum_j \nabla_\theta \ell_\theta^{(j)}$: usually $O(mn)$
  - What if we use really large datasets? (“big data”)
  - What if we learn from data streams? (more data keeps coming in...)

Stochastic / Online Gradient Descent

- Estimate $\nabla_\theta \mathscr{L}_\theta$ fast on a sample of data points
- For each data point:

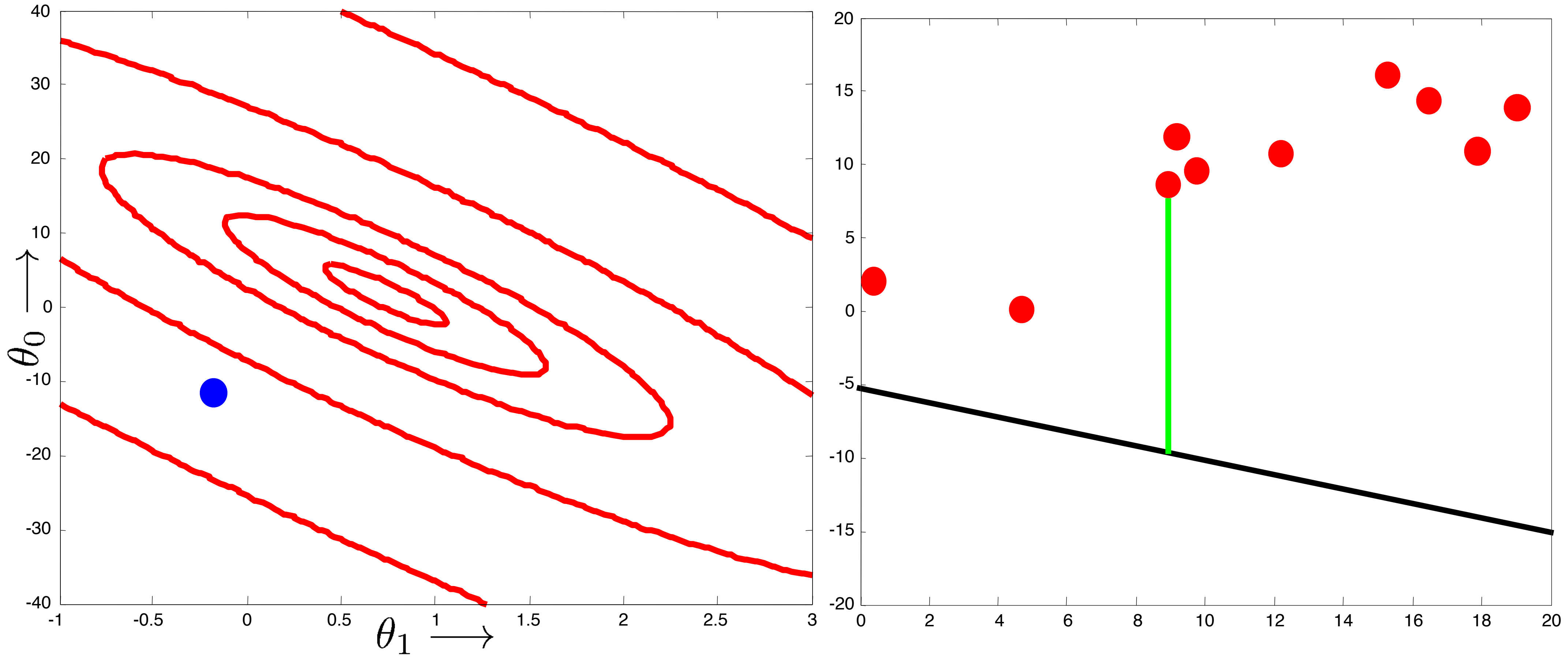
$$\nabla_\theta \mathscr{L}_\theta(x^{(j)}, y^{(j)}) = \nabla_\theta (y^{(j)} - \theta^\intercal x^{(j)})^2 = -2(y^{(j)} - \theta^\intercal x^{(j)})(x^{(j)})^\intercal$$

- This is an unbiased estimator of the gradient, i.e. in expectation

$$\mathbb{E}_{j \sim \text{Uniform}(1,\ldots,m)}[\nabla_\theta \mathscr{L}_\theta^{(j)}] = \frac{1}{m} \sum_j \nabla_\theta \mathscr{L}_\theta^{(j)} = \nabla_\theta \mathscr{L}_\theta(\mathscr{D})$$

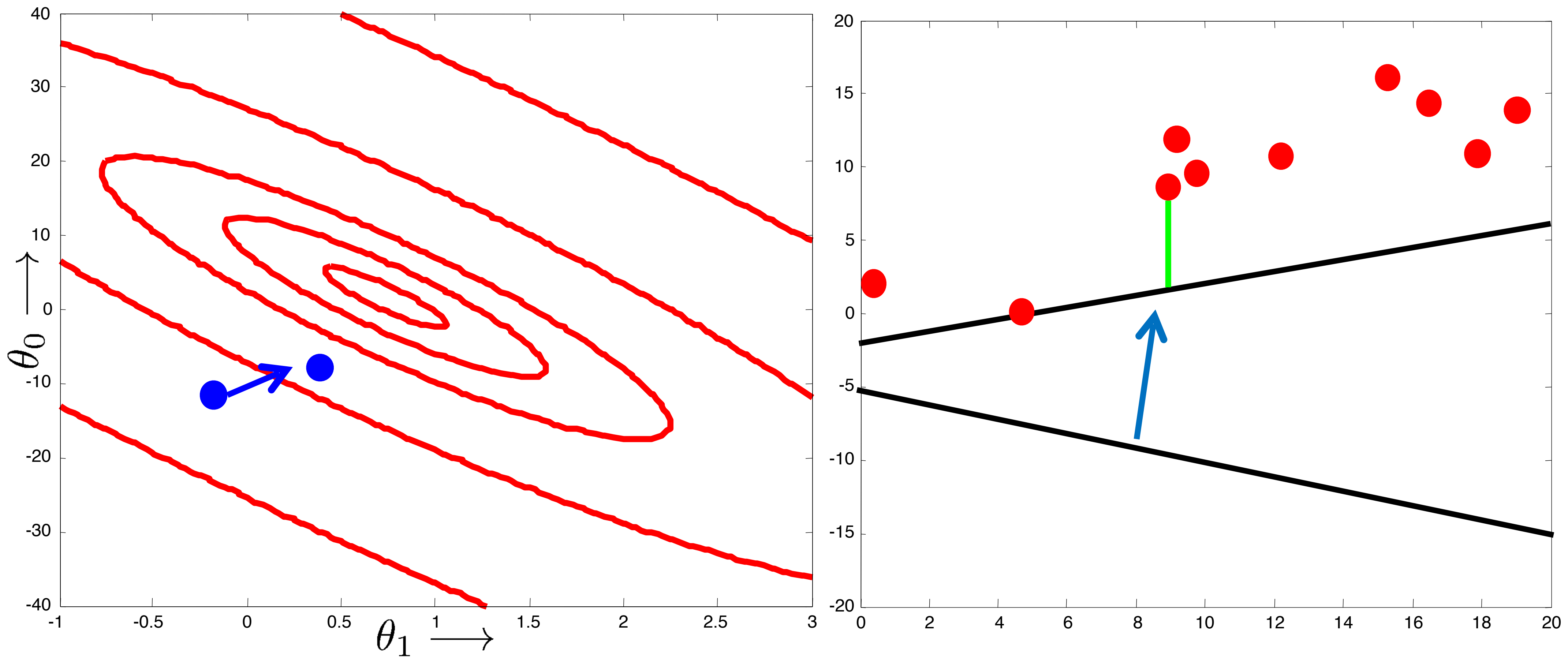
- $\nabla_\theta \mathscr{L}_\theta(\mathscr{D})$ is already a noisy unbiased estimator of true gradient $\mathbb{E}_{x,y \sim p}[\nabla_\theta \mathscr{L}_\theta(x, y)]$
  - SGD is even more noisy

Stochastic Gradient Descent



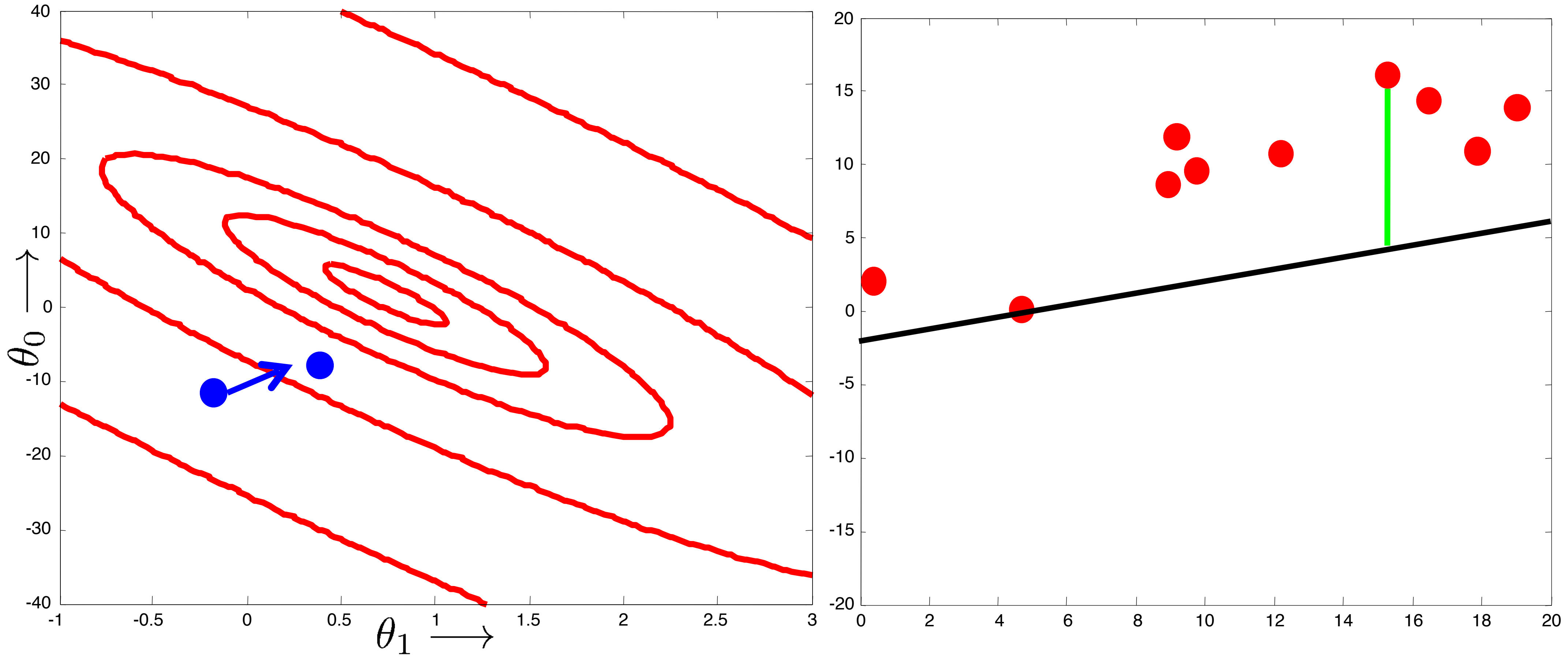
- Initialize θ
- Repeat:
  - Sample $j \sim \mathrm{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no average improvement in $\mathscr{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



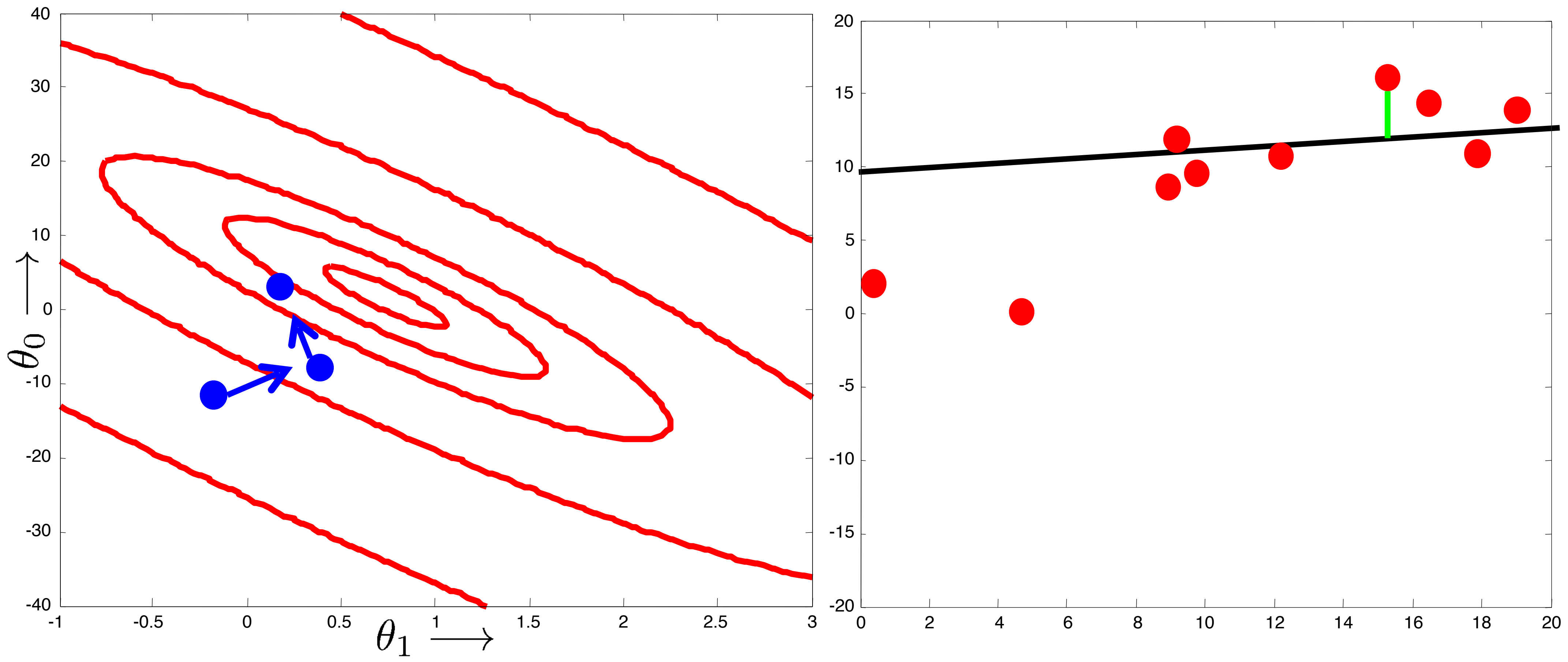
- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no average improvement in $\mathscr{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



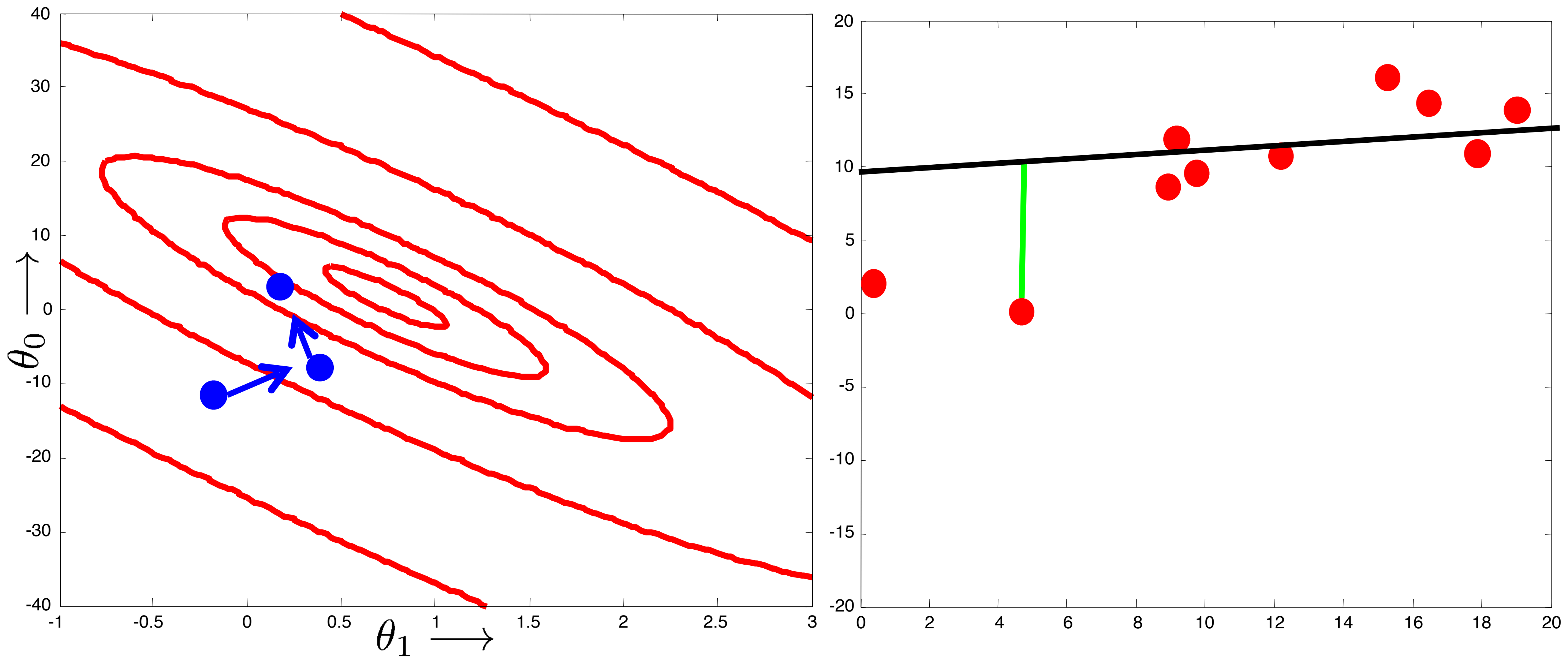
- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no <u>average</u> improvement in $\mathcal{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



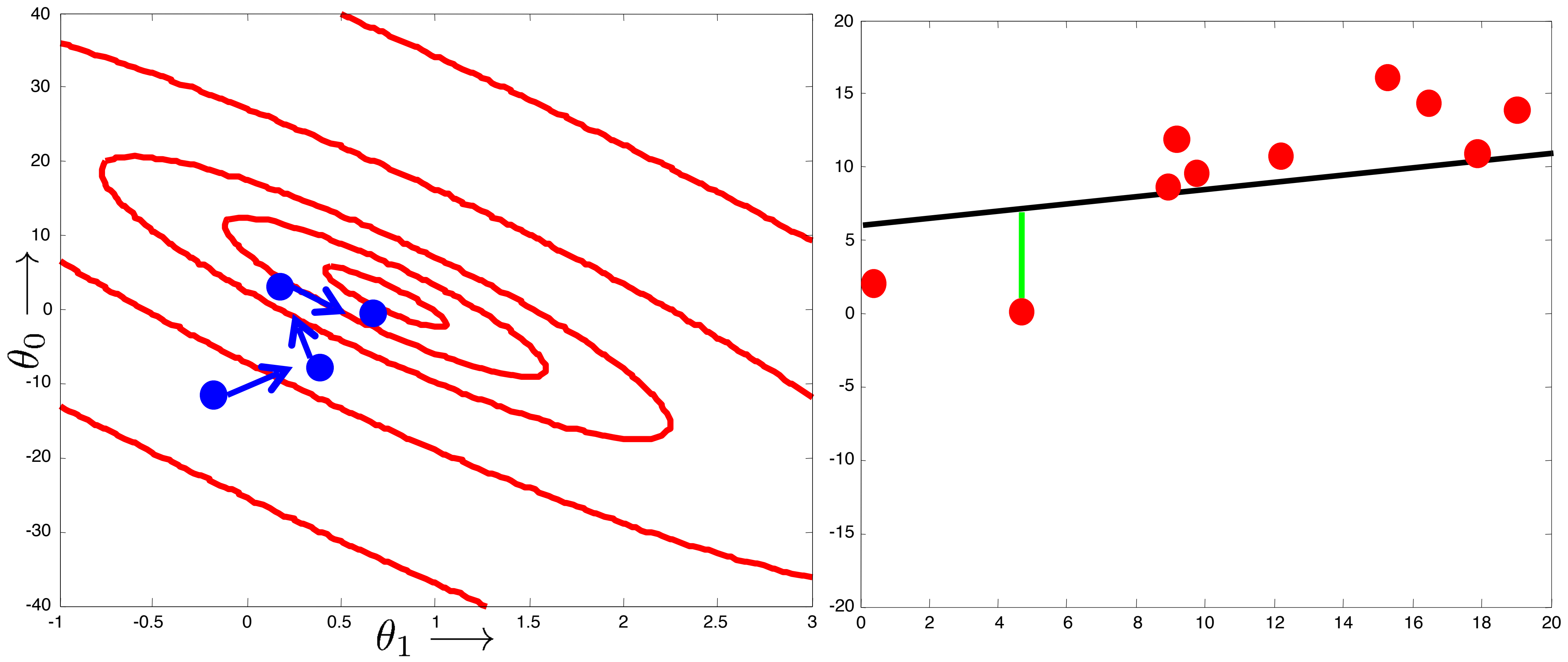
- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no <u>average</u> improvement in $\mathcal{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no average improvement in $\mathcal{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent

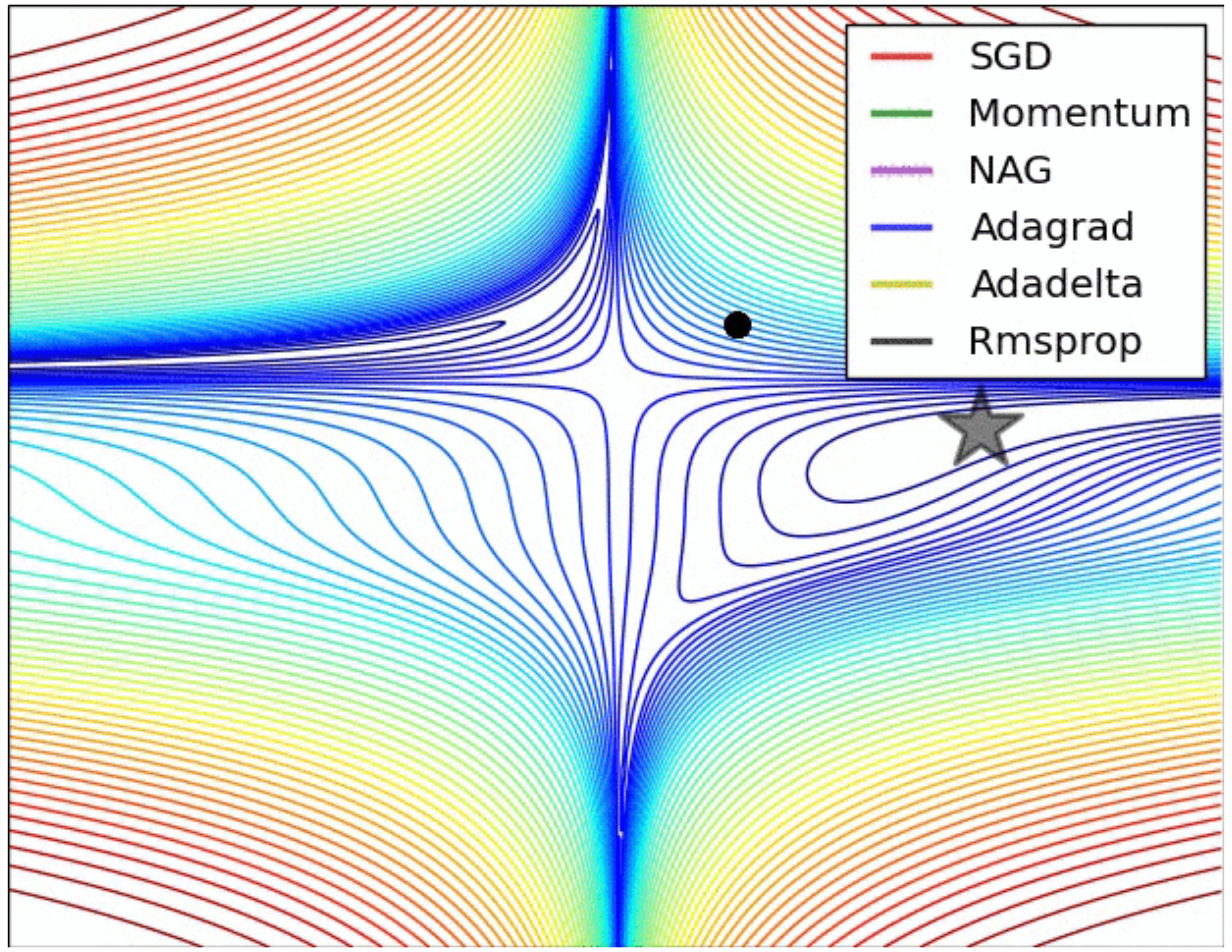


- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no average improvement in $\mathscr{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent: considerations

- Benefits:
  - Each gradient step is faster
  - Don't wait for all data with same θ, improve θ "early and often"
  - Arguably the most important optimization algorithm nowadays
- Drawbacks:
  - May not actually descend on training loss
  - Stopping conditions may be harder to evaluate
- Mini-batch updates: draw $b \ll m$ data points
  - $\operatorname{var} \nabla_\theta \mathscr{L}_\theta(\text{batch}) = \operatorname{var} \frac{1}{b} \sum_{j \in \text{batch}} \nabla_\theta \mathscr{L}_\theta^{(j)} = \frac{1}{b} \operatorname{var} \nabla_\theta \mathscr{L}_\theta(\text{point})$
  - Variance increases the smaller the batch size
    - Generally bad, but can help overcome local minima / saddle points

Advanced gradient-based methods



- Momentum
  - Gradient is like velocity in parameter space
    - Previous gradients still carry momentum
  - Smoothens SGD path
  - Effectively averages gradients over steps, reduces variance
- Preconditioning
  - Scale and rotate loss landscape to make it nicer
  - E.g., multiply by inverse Hessian (as in Newton's method)

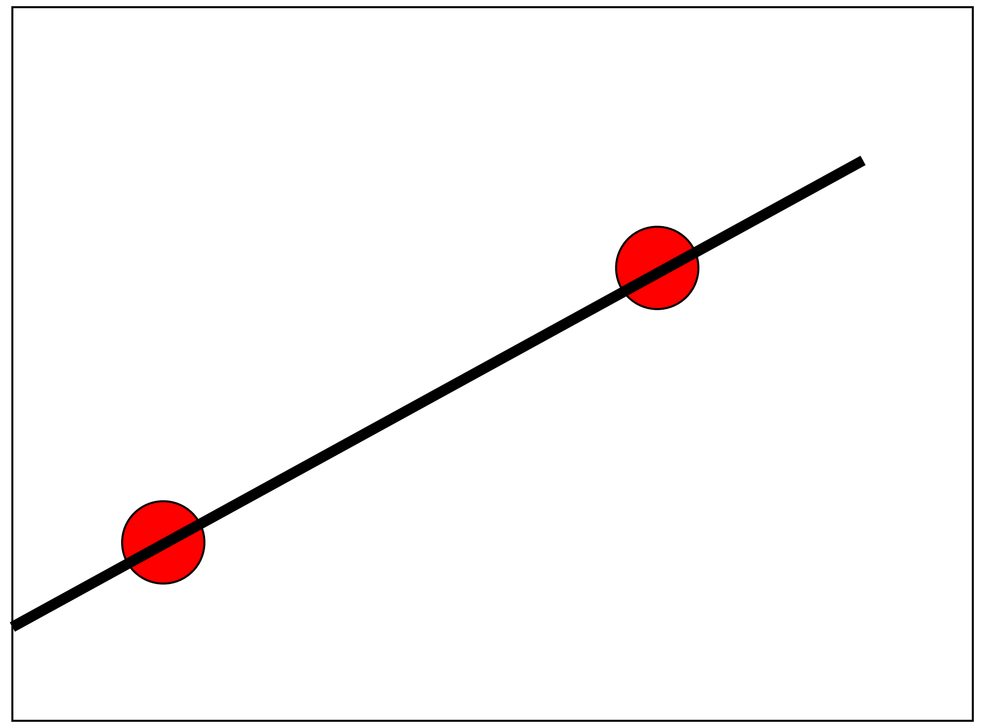
Today's lecture

Stochastic Gradient Descent

Least Squares

Polynomial regression

Minimizing MSE

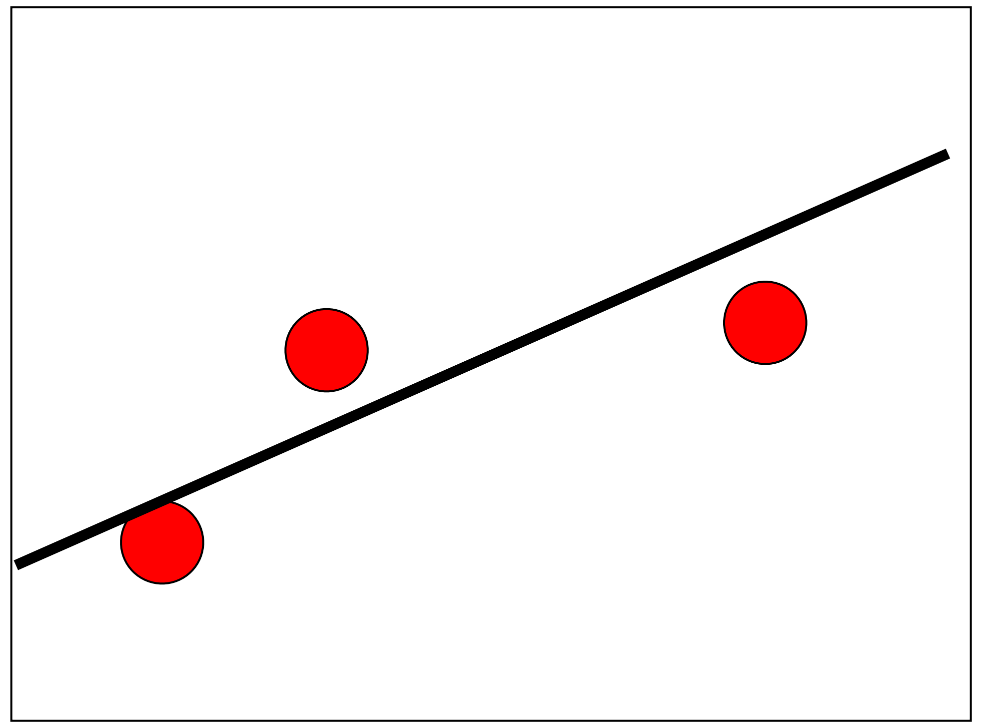


- Consider a simple problem
  - One feature, two data points $x^{(1)}, x^{(2)}$
  - Two unknowns θ_0, θ_1
  - Two equations: $\theta_0 + \theta_1 x^{(1)} = y^{(1)}$ $\qquad \theta_0 + \theta_1 x^{(2)} = y^{(2)}$
- Can solve this system directly: $y = \theta^\intercal X \implies \theta^\intercal = yX^{-1}$
- Generally, X may not have an inverse; e.g., $m > n$
- There may also be training loss, no θ achieves equality of y to $\theta^\intercal X$

Least Squares

- The minimum is achieved when the gradient is 0

$$\nabla_\theta \mathcal{L}_\theta = -\frac{2}{m}(y - \theta^\intercal X)X^\intercal = 0$$
$$\theta^\intercal X X^\intercal = y X^\intercal$$
$$\theta^\intercal = y X^\intercal (X X^\intercal)^{-1}$$



- $XX^\intercal$ is invertible when X has linearly independent rows = features
- $X^\dagger = X^\intercal (XX^\intercal)^{-1}$ is the Moore-Penrose pseudo-inverse of X
  - $X^\dagger = X^{-1}$ when the inverse exists
  - Can define $X^\dagger$ via Singular Value Decomposition (SVD) when $XX^\intercal$ isn't invertible
- $\theta^\intercal = y X^\dagger$ is the Least Squares fit of the data (X, y)

Linear regression in NumPy

- Linear regression with MSE: $\min_\theta \frac{1}{m}\|y - \theta^\intercal X\|^2$

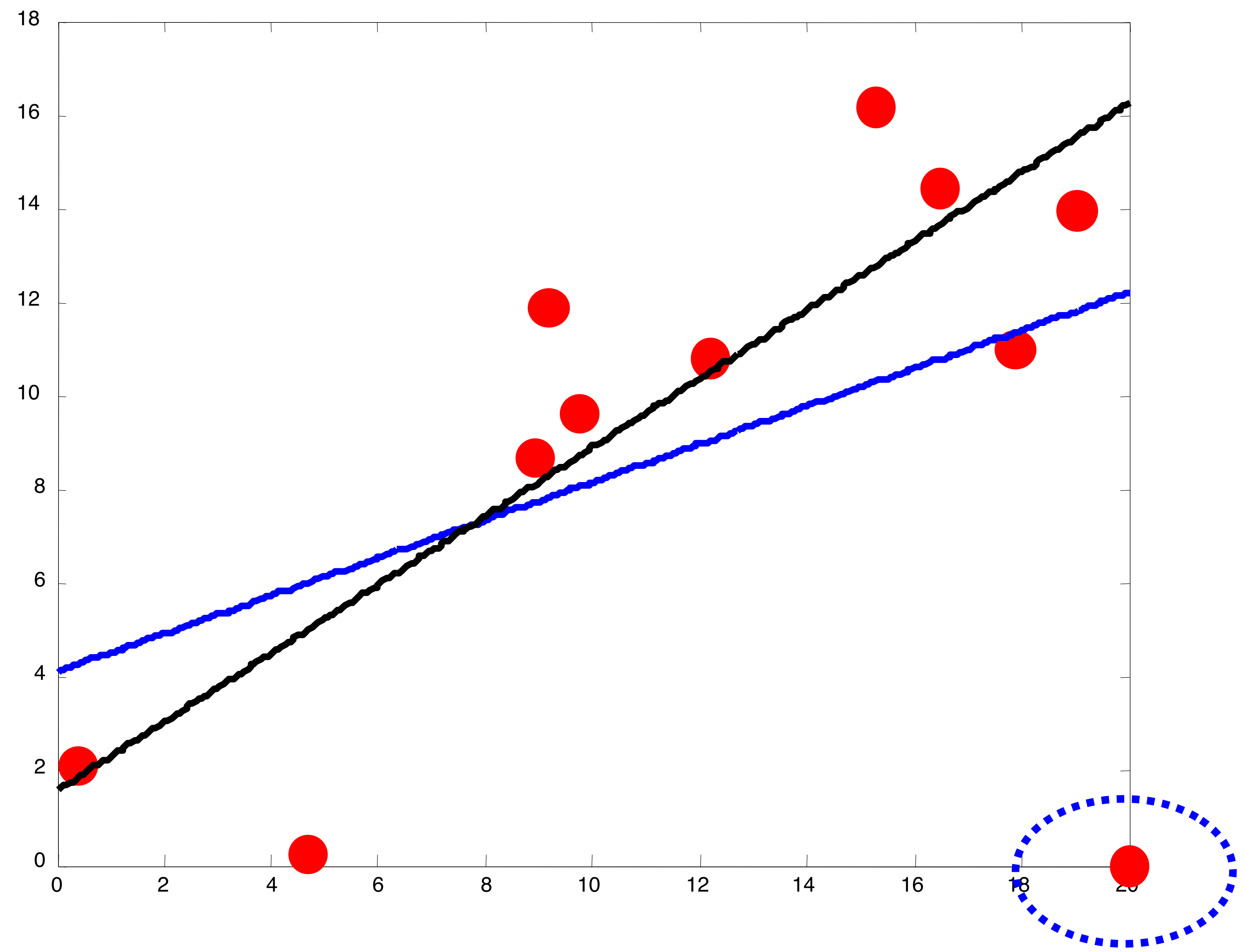
$$\theta^\intercal = yX(XX^\intercal)^{-1} = yX^\dagger$$

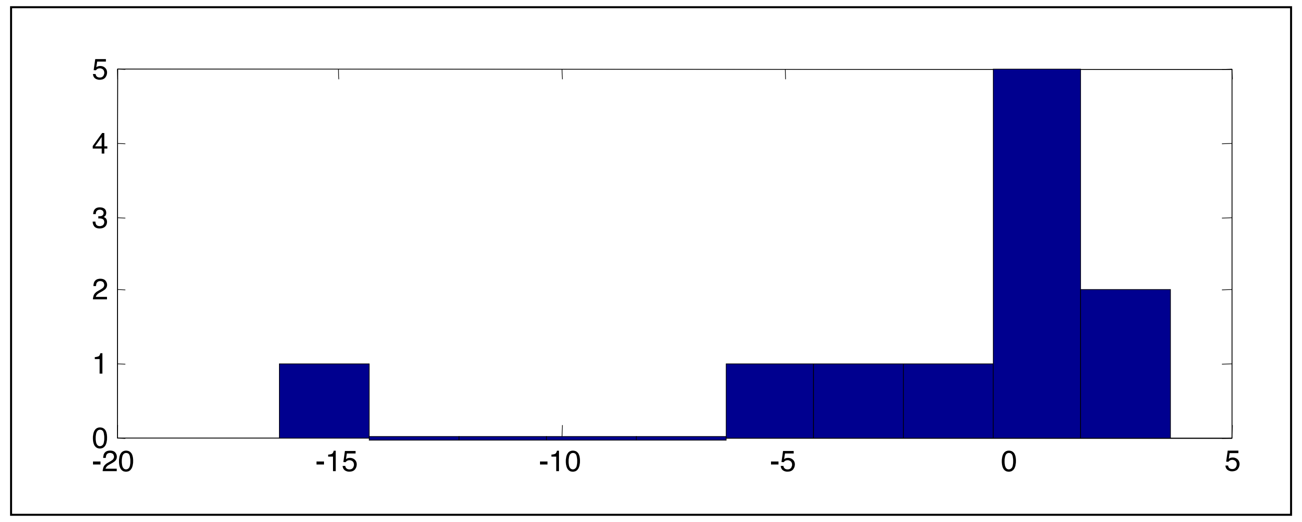
```
# Solution 1: the long way
theta = (y @ X @ np.linalg.inv(X @ X.T)).T

# Solution 2: pseudo-inverse
theta = (y @ np.linalg.pinv(X)).T

# Solution 3: Least Squares solver
theta = np.linalg.lstsq(a=X.T, b=y.T)
```

- Least Squares: approximate $Az = b$ by $\min_z \|Az - b\|^2$

MSE and outliers

- MSE is sensitive to outliers





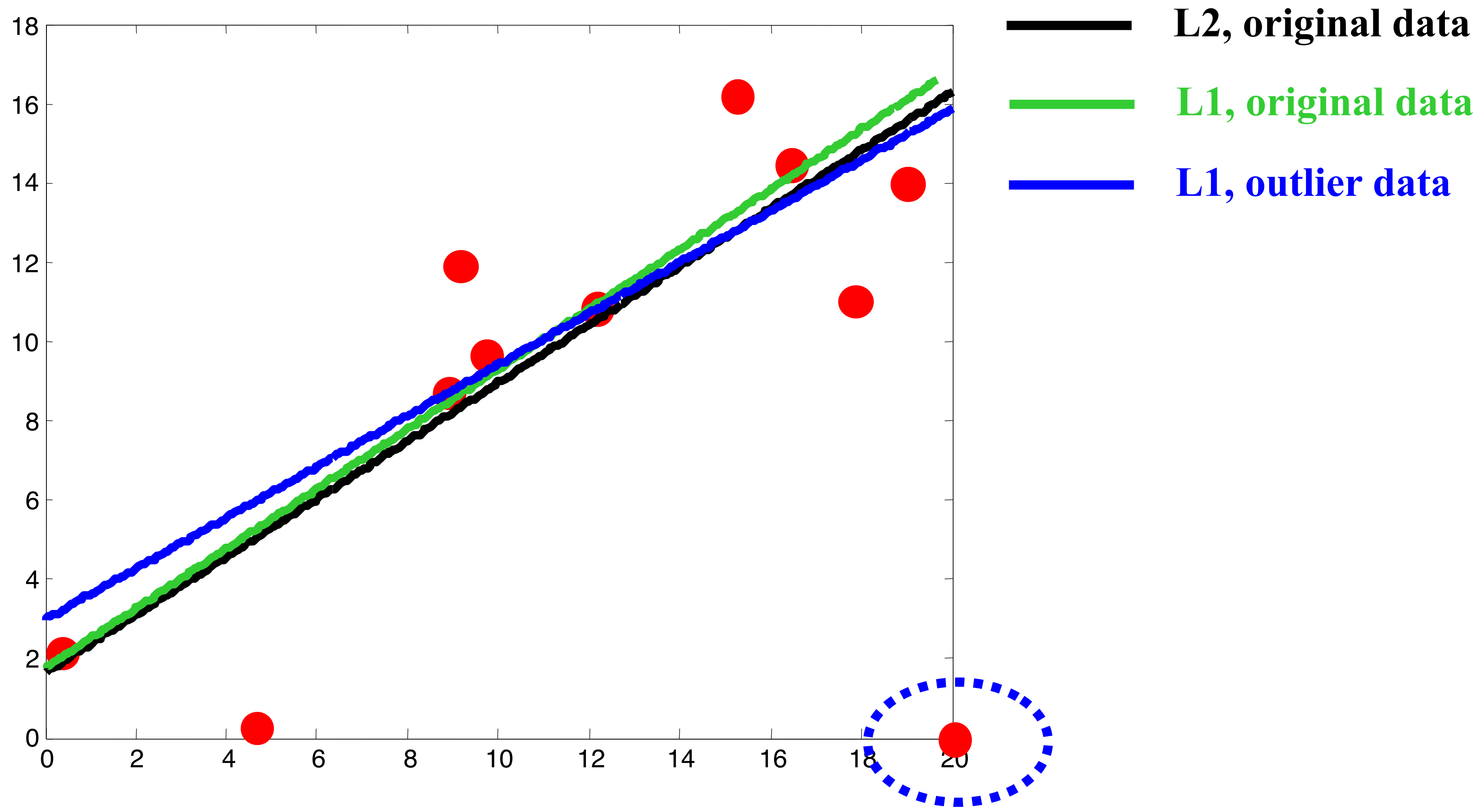
- Square error 16^2 throws off entire optimization

Mean Absolute Error (MAE)

- MSE uses the L_2 norm of the error $\|y - \theta^\intercal X\|_2^2 = \sum_j (y - \theta^\intercal X)^2$

- What if we use the L_1 norm $\|y - \theta^\intercal X\|_1 = \sum_j |y - \theta^\intercal X|$?

    - Mean Absolute Error (MAE): $\frac{1}{m} \sum_j |y - \theta^\intercal X|$



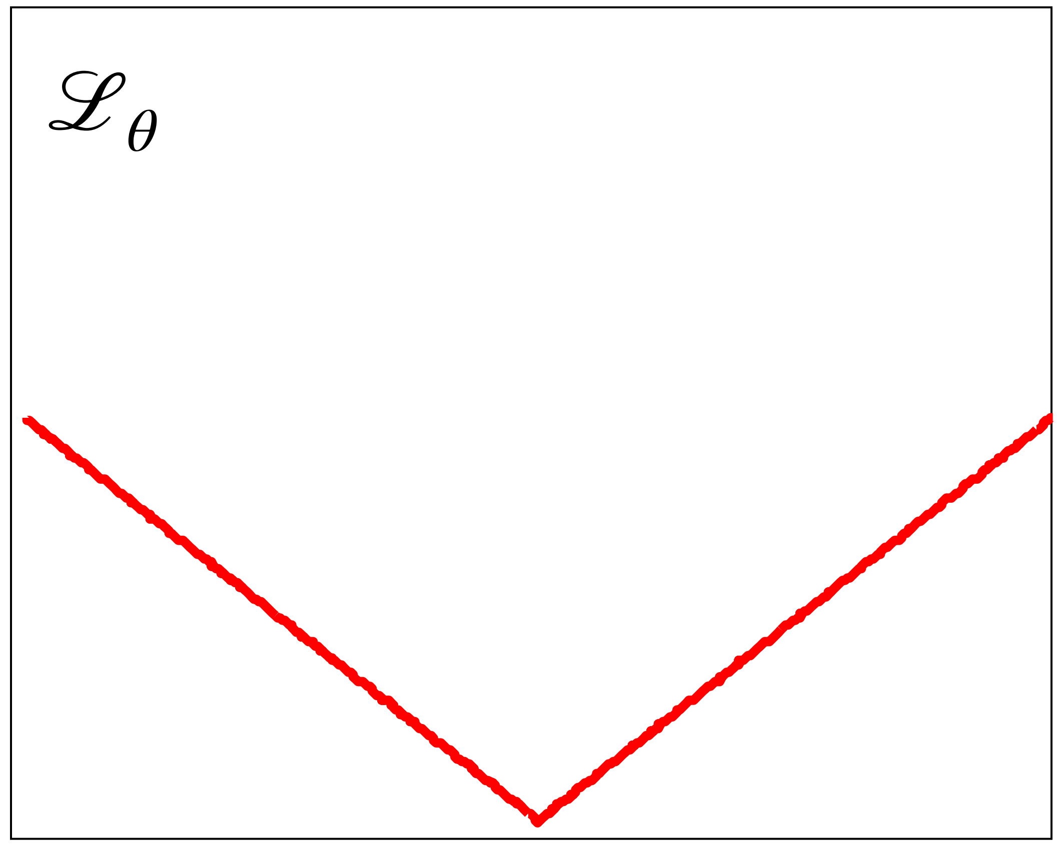
Minimizing MAE

- The absolute operator isn't differentiable
  - But assume no data point has 0 error

$$\nabla_\theta \frac{1}{m} \sum_j |y - \theta^\intercal X| = \frac{1}{m} \left(\sum_{j:\ y^{(j)} < \theta^\intercal x^{(j)}} x^{(j)} - \sum_{j:\ y^{(j)} > \theta^\intercal x^{(j)}} x^{(j)} \right) = 0$$

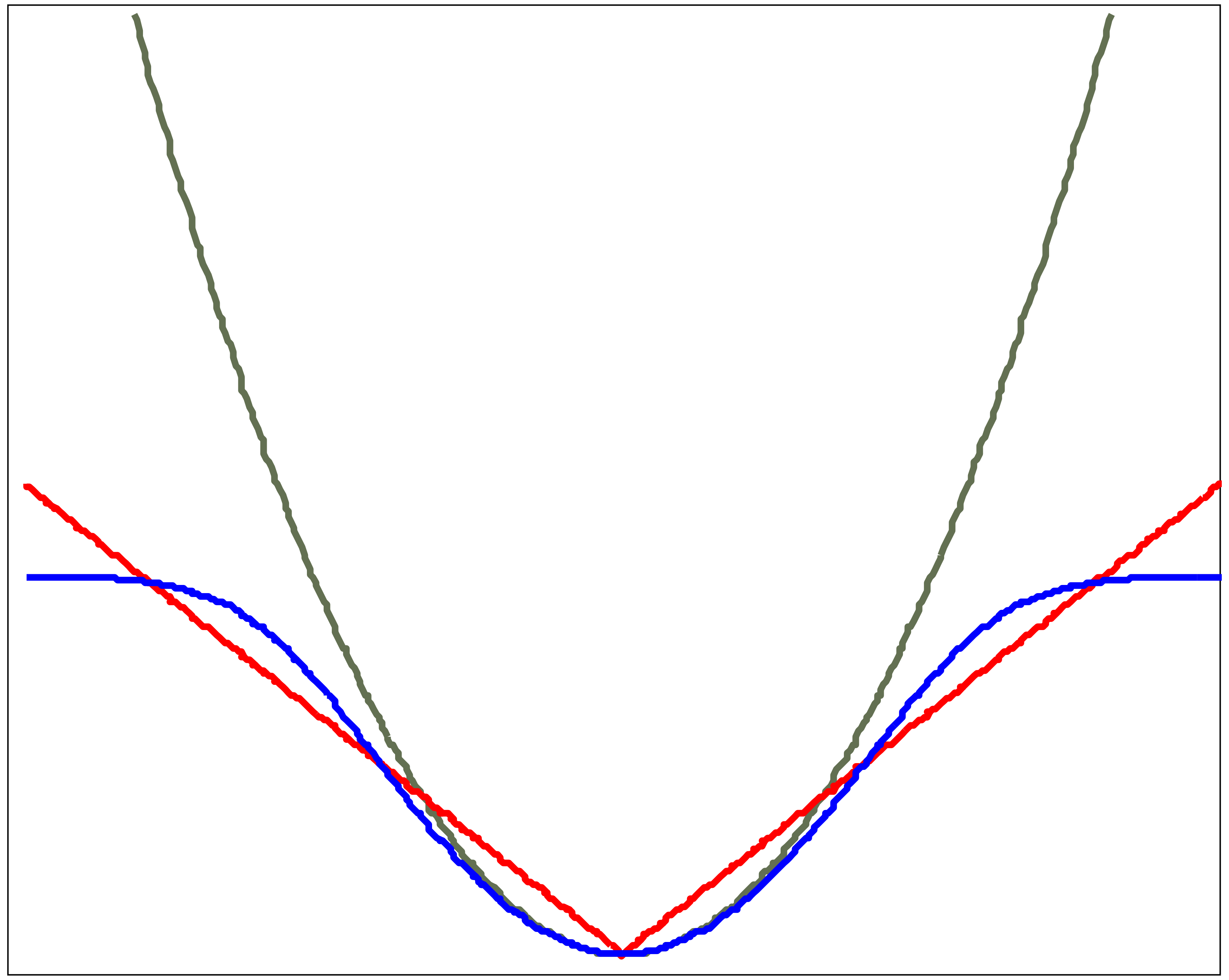
$$\sum_{j:\ y^{(j)} < \theta^\intercal x^{(j)}} x^{(j)} = \sum_{j:\ y^{(j)} > \theta^\intercal x^{(j)}} x^{(j)}$$

- Can be solved with Linear Programming
- Without features (best constant fit for y): median
  - With MSE: mean — more sensitive to outliers



Other loss functions

- MSE: $\ell(y, \hat{y}) = (y - \hat{y})^2$
- MAE: $\ell(y, \hat{y}) = |y - \hat{y}|$
- Should loss of large errors saturate?
  - $\ell(y, \hat{y}) = c - \log(\exp(-(y - \hat{y})^2) + c)$



- Most loss functions cannot be optimized in close form
  - Gradient descent is a general algorithm for differentiable parametrization and loss

Today's lecture

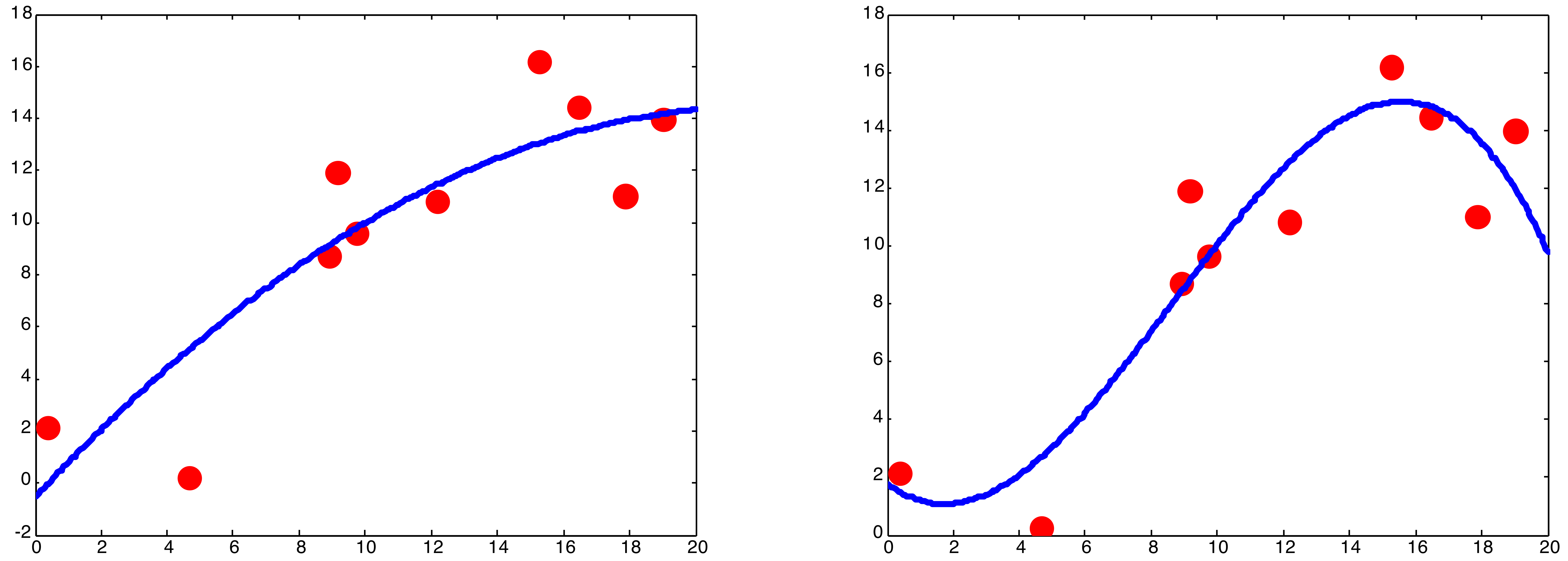
Stochastic Gradient Descent

Least Squares

Polynomial regression

Polynomial regression

- Some data cannot be explained by linear regression
  - A higher-order polynomial may be a better fit



Polynomial regression

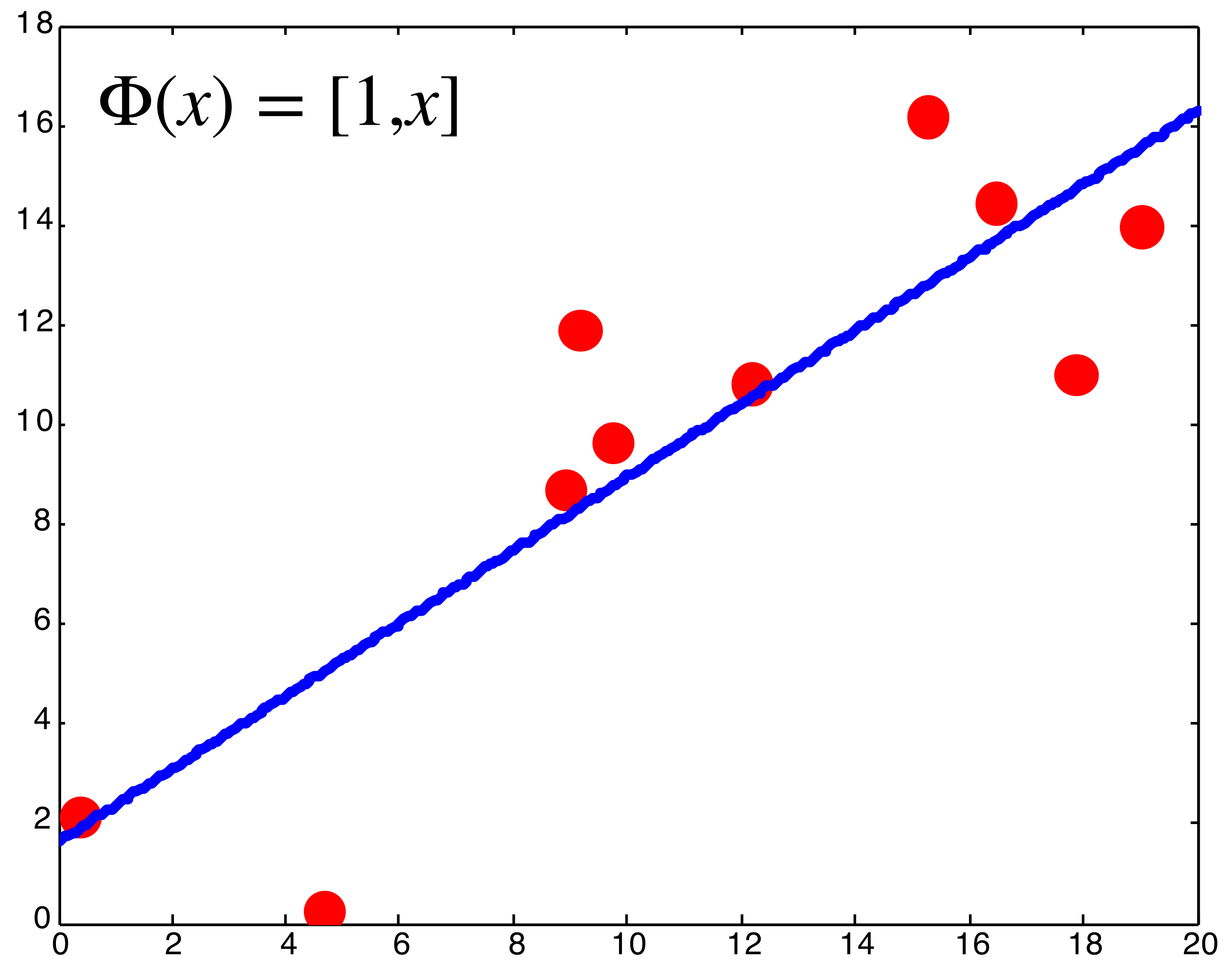
- Consider a polynomial in a single feature x

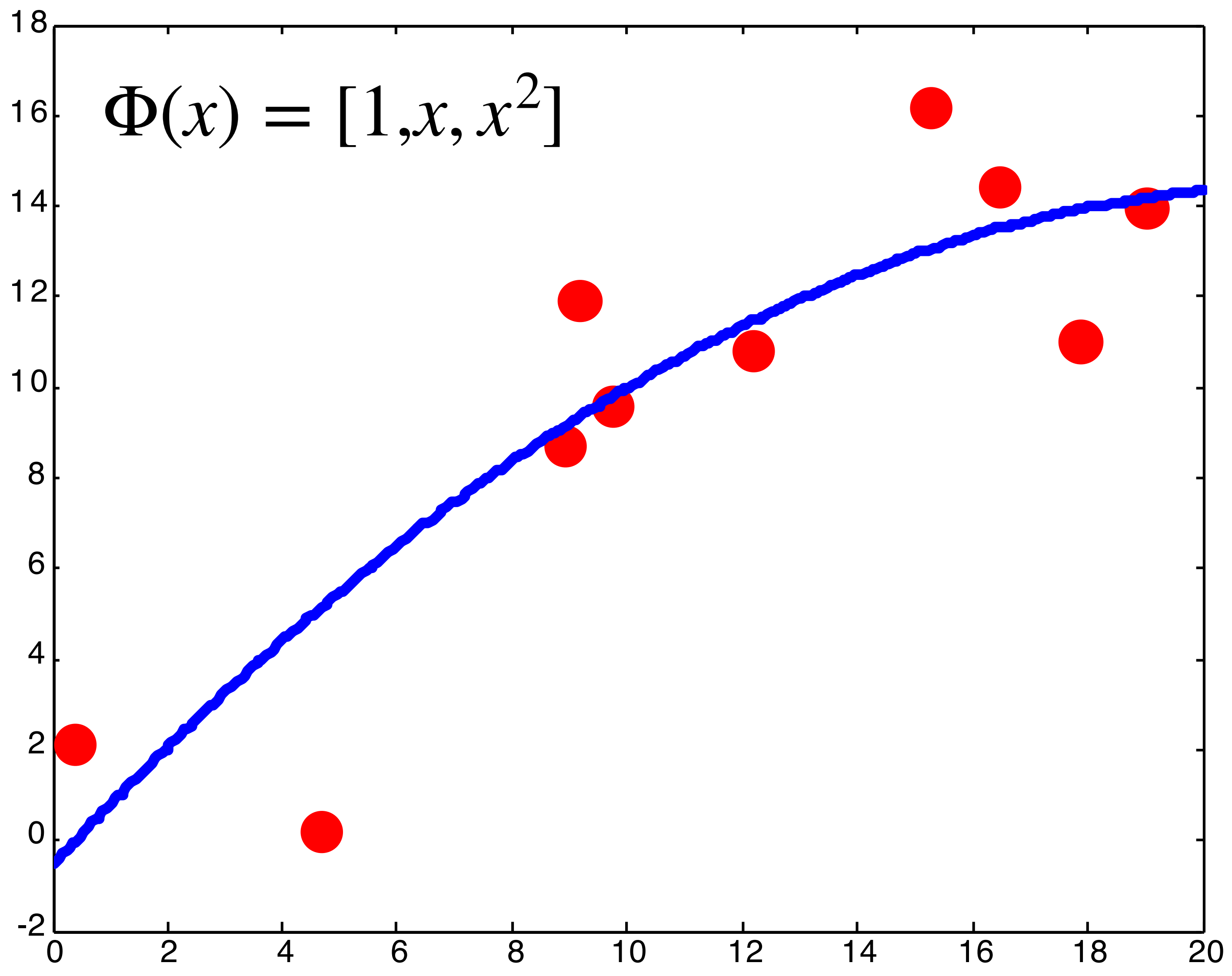
$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots$$

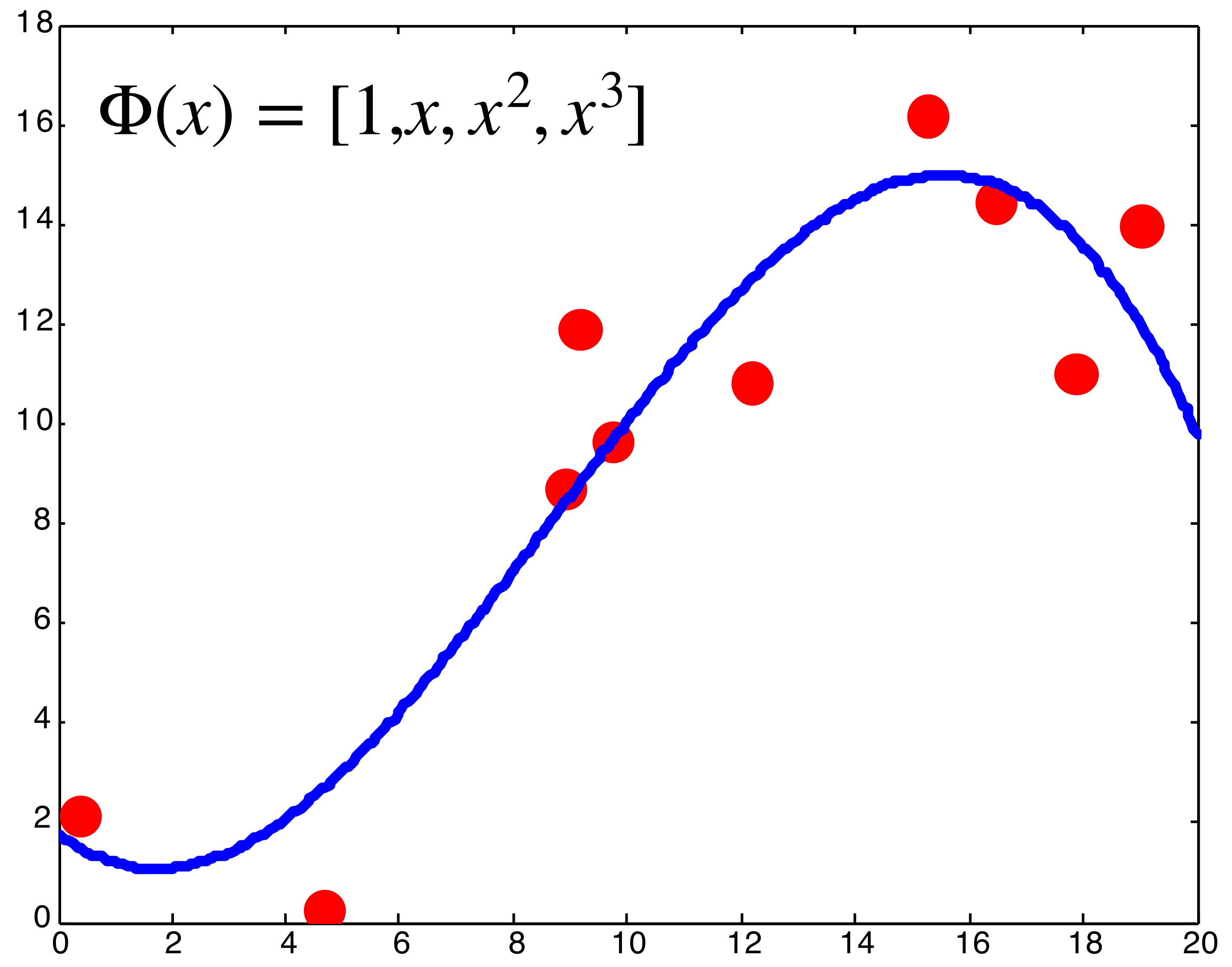
- Can we reduce this to something we already know?
  - Think of higher-order terms $x^2, x^3, \dots$ as new features
- $\mathscr{D} = \{(x^{(j)}, y^{(j)})\} \implies \{([x^{(j)}, (x^{(j)})^2, (x^{(j)})^3, \dots], y^{(j)})\}$
  - Denote $\Phi(x) = [x, x^2, x^3, \dots]$
- Perform linear regression with $\hat{y} = \theta^\intercal \Phi(x)$

Polynomial regression

- Fit the same way as linear regression
  - With more features $\Phi(x)$





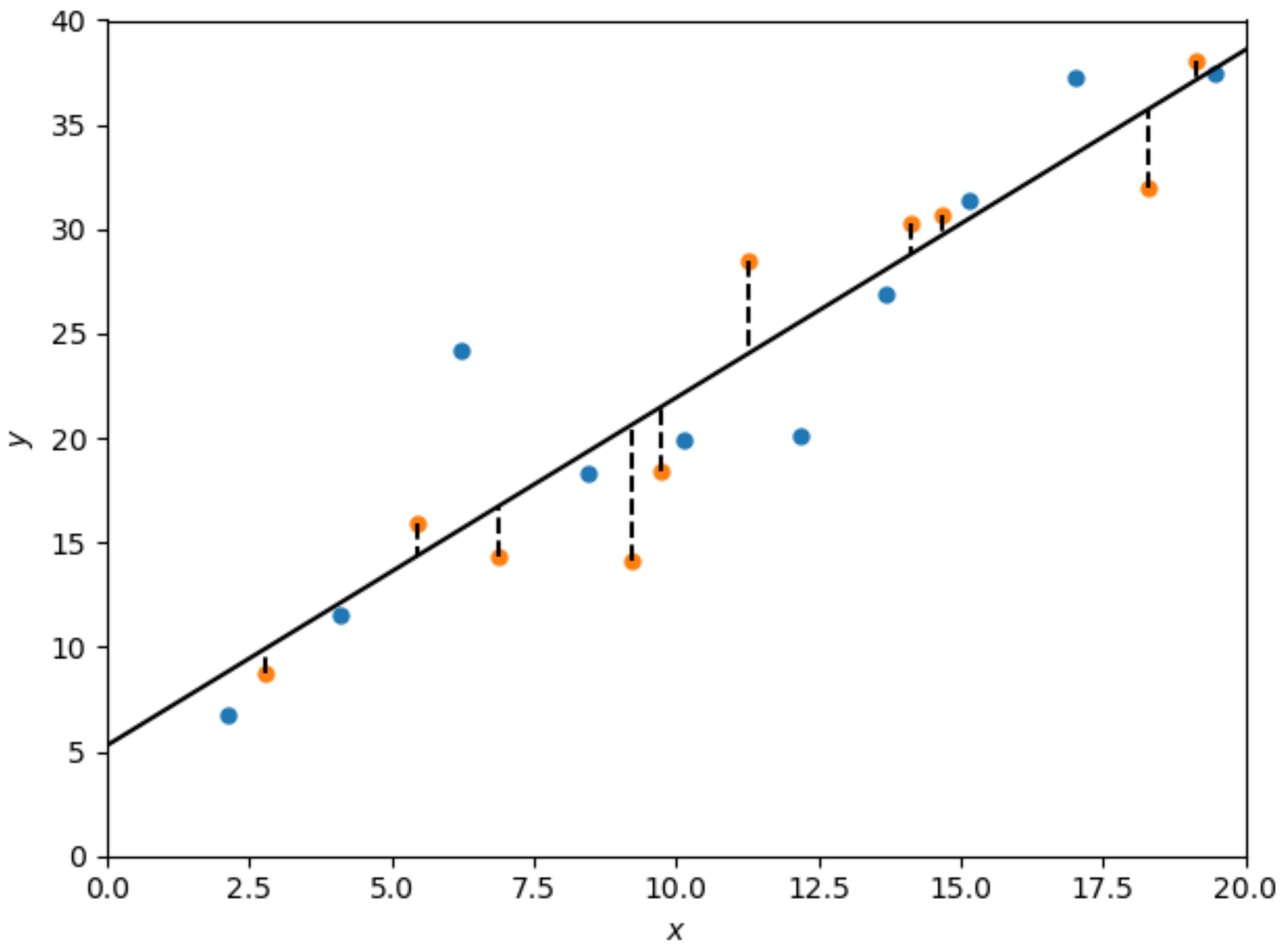


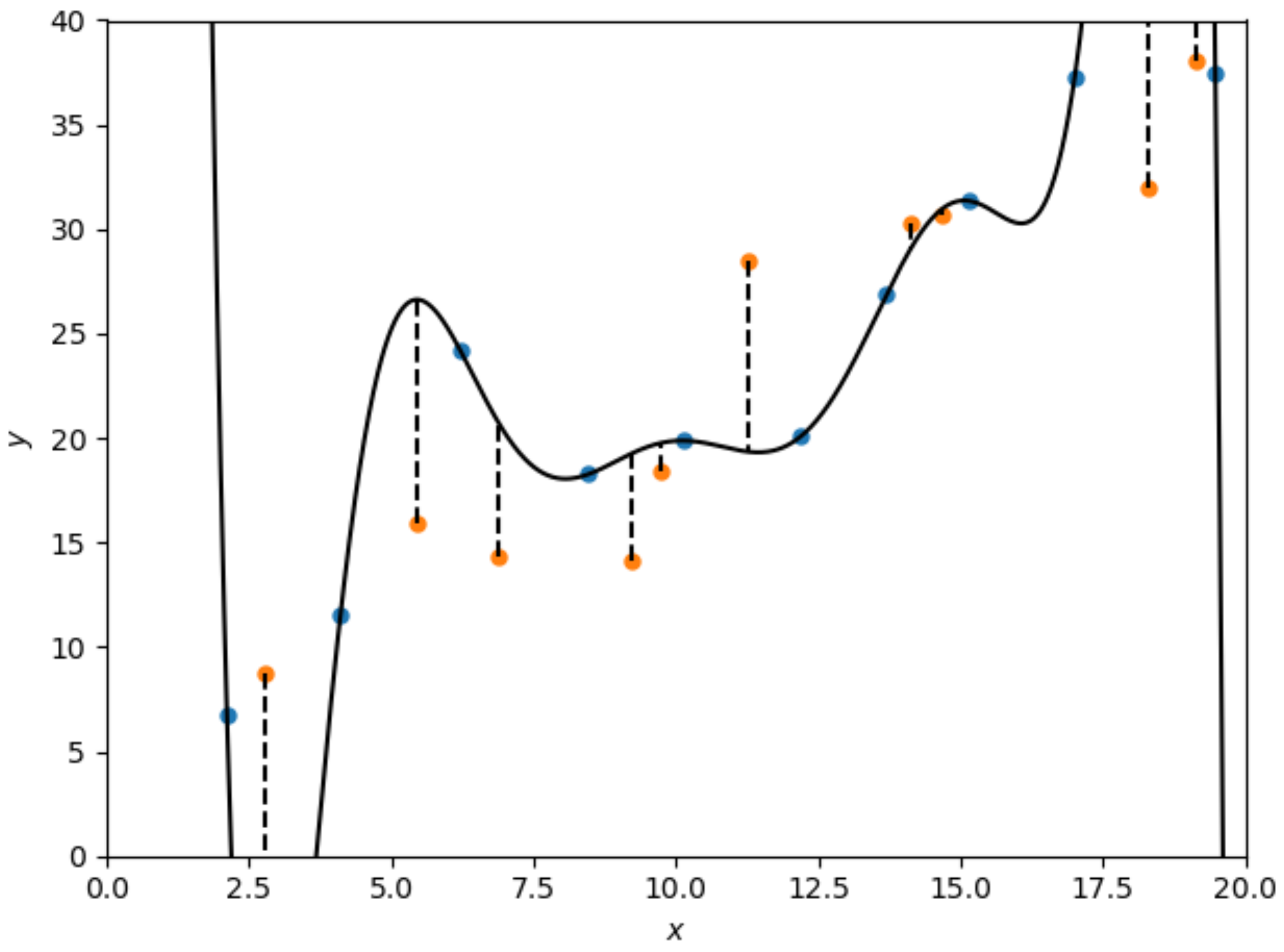
Feature expansion

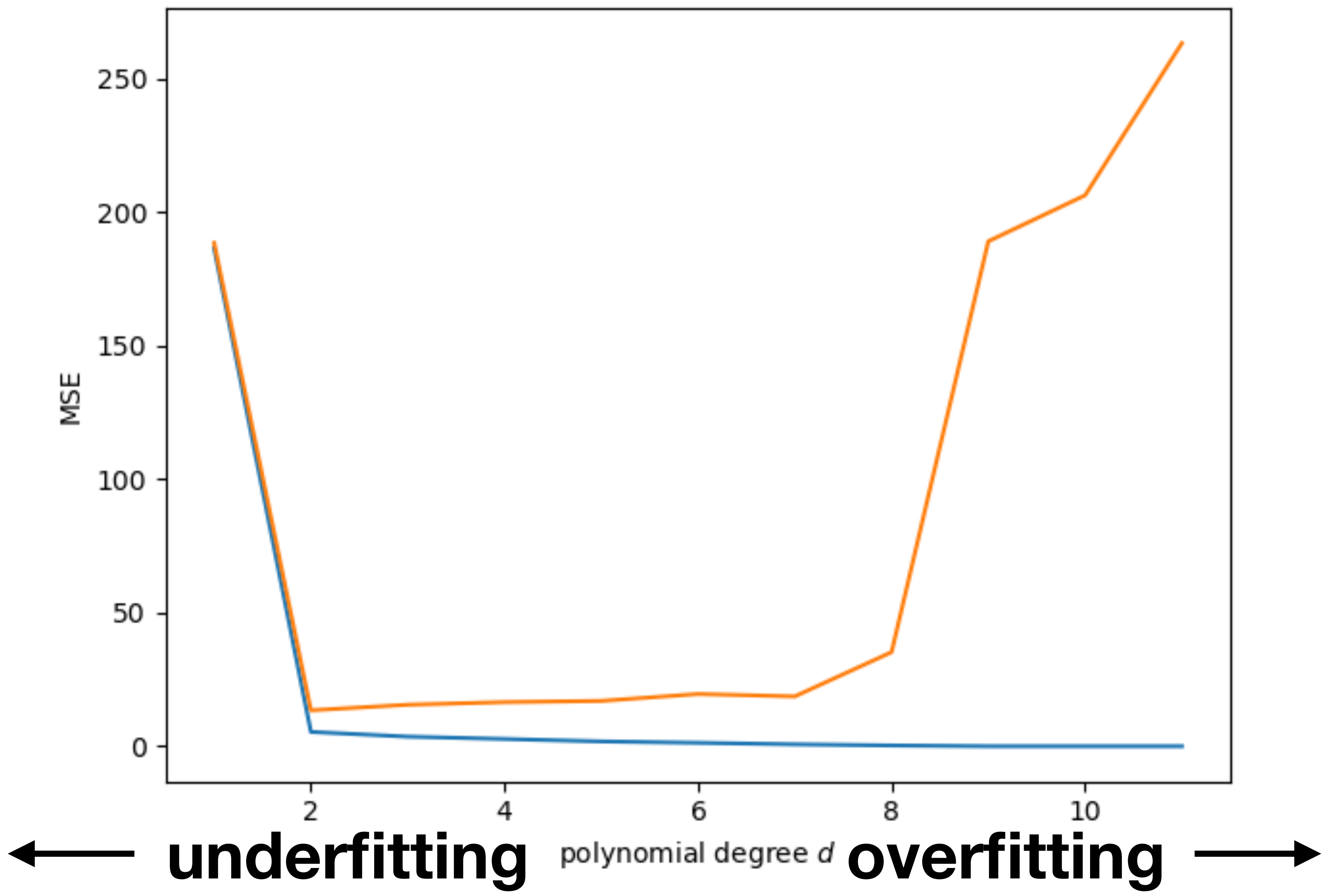
- In principle, can use any features we think are useful
- Instead of collecting more information per data point
  - apply nonlinear transformation to x to get more "linear explainability" of y
- More examples:
  - Cross-terms between features: $x_i x_j$, $x_i x_j x_k$, ...
  - Trigonometric functions: $sin(\omega x + \phi)$
  - Others: $\frac{1}{x}$, $\sqrt{x}$, ...
- Linear regression = linear in θ, the features can be as complex as we want

How many features to add?

- The more features we add, the more complex the model class
- Learning can always fall back to simpler model with $\theta_4 = \theta_5 = \cdots = 0$
- But generally it won't, it will overfit
  - Better training data fit, worse test data fit







Inductive bias

- Inductive bias = assumptions we make to generalize to data we haven't seen
  - 10 data points suggest 9-degree polynomial, but we're “biased” towards linear
  - Examples: polynomials, smooth functions, neural network architecture, etc.
- Without any assumptions, there is no generalization
  - “Anything is possible” in the test data
- Occam's razor: prefer simpler explanations of the data

Logistics

assignments

- Assignment 2 to be published soon

project

- Project guidelines to be published soon
- Team rosters due next Thursday, Jan 28